



Ver. 1

Survey

API

---

CheckMarket's application programming interface (API) allows developers to hook into CheckMarket and integrate its functionality into internal applications.

**Documentation**

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b> .....	<b>2</b>
<b>INTRODUCTION</b> .....	<b>4</b>
<b>1 RESTFUL API</b> .....	<b>5</b>
<b>1.1 Introduction</b> .....	<b>5</b>
<b>1.2 Terminology</b> .....	<b>5</b>
1.2.1 Resource .....	5
1.2.2 Resource identifier .....	5
1.2.3 Attribute .....	5
<b>1.3 Overview</b> .....	<b>5</b>
1.3.1 HTTP-request requirements .....	5
1.3.1.1 URI .....	5
1.3.1.2 Authentication .....	6
1.3.1.3 Method .....	6
1.3.2 Response and Status Codes .....	7
1.3.3 Error messages .....	8
1.3.4 List of Status Codes .....	8
1.3.5 JSONP .....	8
<b>1.4 Functions</b> .....	<b>10</b>
1.4.1 Contacts .....	10
1.4.2 Groups .....	10
1.4.3 Folders .....	10
1.4.4 Surveys .....	11
<b>1.5 Scenarios</b> .....	<b>11</b>
1.5.1 Adding contacts to the address book & panel .....	11
1.5.1.1 Case 1: Adding a single contact, success .....	12
1.5.1.2 Case 2: Adding a single contact, error .....	13
1.5.1.3 Case 3: Adding a single contact, error .....	14
1.5.1.4 Case 4: Adding multiple contacts, success .....	15
1.5.1.5 Case 5: Adding multiple contacts, error .....	17
1.5.2 Adding a Folder to the address book .....	18
1.5.2.1 Case 1: Adding a folder, success .....	18
1.5.2.2 Case 2: Adding a folder, error .....	18
1.5.2.3 Case 3: Updating attributes of a folder .....	19
1.5.3 Adding a Group to the address book .....	19
1.5.3.1 Case 1: Adding a group, success .....	19
1.5.3.2 Case 2: Adding a group, error .....	20
1.5.3.3 Case 3: Updating attributes of a group .....	20
<b>2 SOAP API</b> .....	<b>21</b>
<b>2.1 Introduction</b> .....	<b>21</b>
<b>2.2 Authentication</b> .....	<b>21</b>
<b>2.3 Functions</b> .....	<b>21</b>
2.3.1 Contacts .....	21
2.3.2 Groups .....	22
2.3.3 Folders .....	22
2.3.4 Surveys .....	22

<b>3</b>	<b>ENTITIES .....</b>	<b>23</b>
<b>3.1</b>	<b>Contact .....</b>	<b>23</b>
<b>3.2</b>	<b>ContactOptions .....</b>	<b>24</b>
<b>3.3</b>	<b>Folder .....</b>	<b>25</b>
<b>3.4</b>	<b>Group .....</b>	<b>26</b>
<b>3.5</b>	<b>Survey .....</b>	<b>27</b>
<b>3.6</b>	<b>Report .....</b>	<b>28</b>
<b>3.7</b>	<b>SuccessReport .....</b>	<b>28</b>
<b>3.8</b>	<b>FailedReport .....</b>	<b>29</b>
<b>3.9</b>	<b>ContactReport .....</b>	<b>29</b>

## INTRODUCTION

CheckMarket's application programming interface (API) allows developers to hook into CheckMarket and integrate its functionality into internal applications.

**You can integrate CheckMarket data gathering technology with your:**

- CRM
- HR
- ERP
- Call Center
- Customer Service

or other key systems to analyze and compare all kinds of information including customer, employee, partner, and member attitudinal data.

**CheckMarket's new API enables users to:**

- Automatically send surveys after each pre-defined event, such as a sale or customer service contact.
- Create relevant survey questions based on the individual customer's profile stored in your company's database.
- Integrate historical customer survey responses into the company's CRM.

To make integration even easier, the CheckMarket API SOAP interface supports a growing number of languages including Java, .NET, PHP and Ruby.

In addition, SOAP request/response messages enable users to identify errors on-the-fly without sending requests to CheckMarket's support team.

Moreover, CheckMarket has a dedicated team of developers with extensive experience integrating CheckMarket software with all kinds of enterprise applications.

# 1 RESTFUL API

## 1.1 Introduction

Communication with CheckMarket's RESTful API happens via simple HTTP requests to <https://api.checkmarket.com>.

CheckMarket's RESTful API adheres to the principles of REST architecture. More information about this architecture can be found [here](#).

## 1.2 Terminology

This section describes the basic terminology within the RESTful context.

### 1.2.1 Resource

A resource is the RESTful representation of an entity within CheckMarket's system. A contact or a folder in the address book are examples of resources.

Resources can be accessed via a [URI](#). An example URI to retrieve a collection of resources could look like this: `api.checkmarket.com/{resource}`.

Every resource has got a *resource identifier*. This is how a single resource can be accessed. The resource identifier is appended to the end of the URI of a specific resource. Example: `api.checkmarket.com/{resource}/{id}`. 'id' represents the Id attribute of a specific resource.

### 1.2.2 Resource identifier

An integer value to address a specific resource.

### 1.2.3 Attribute

A property of a resource. Resource 'Contact' could have attribute 'FirstName'.

## 1.3 Overview

### 1.3.1 HTTP-request requirements

To access a resource 4 things are needed:

- URI;
- Format (data-interchange format);
- Method (action);
- API key (authentication)




#### 1.3.1.1 URI

The most complete Unique Resource Identifier looks like this:

`https://api.checkmarket.com/{version}/{resource}/{id}`

The URI contains of three parts.

-  `api.checkmarket.com` is the basic URL;

-  {version} targets a specific version of the API;
-  {resource} defines a resource;
-  {id} is the resource identifier (the 'Id' attribute of a resource)

Example 1: A contact with Id 620: <https://api.checkmarket.com/1/contact/620>

Example 2: A collection of contacts: <https://api.checkmarket.com/1/contacts>

The version number can be left out. This way you'll be communicating with the latest version of the API. *Note that targeting a specific version of the API is recommended to avoid breaking changes.*

CheckMarket's RESTful API supports XML and JSON as data-interchange format. Data can be retrieved and accessed in both formats. The format is specified in the Content-Type HTTP header.

Valid HTTP headers:

- Content-Type: application/xml
- Content-Type: application/json

#### 1.3.1.2 Authentication

Authentication happens via a unique key, the API key, which is connected to your account. The API key can be found in the member section under "my account". The key should be passed along with every request under the X-CheckMarket-Key HTTP header.

Example: X-CheckMarket-Key: 5x4f6z84fz67z9z45p

#### 1.3.1.3 Method

The HTTP method will reflect the action on a resource. Supported HTTP methods are:

- GET – to retrieve a resource
- POST – to create a resource
- PUT – to update a resource
- DELETE – to delete a resource

These methods are often respectively associated with the READ, CREATE, UPDATE, DELETE operations associated with database technologies.

Example GET 1: Retrieve a collection of resources: **GET /{resource}**

Example: [GET /contacts/1](https://api.checkmarket.com/1/contacts)

Example GET 2: Retrieve a specific resource: **GET /{resource}/{id}**

Example: [GET /contact/620](https://api.checkmarket.com/1/contact/620)

Example POST: Creating a resource: **POST /{resource}**

Example: [adding a folder to the address book](#)

#### **Request:**

Method	POST /folder
--------	--------------

Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<Folder> <Name>Map 1</Name> <Description>Some classification</Description> </Folder>

Example PUT:Updating a resource: **PUT /{resource}/{id}**Example: updating the Description attribute of a folder in the address book

Method	PUT /folder/857
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<Folder> <Description>Better description</Description> </Folder>

Example DELETE:Deleting a resource: **DELETE /{resource}/{id}**Example: deleting folder 857

Method	DELETE /folder/857
Headers	X-CheckMarket-Key: 5x4f6z84fz67z9z45p

**1.3.2 Response and Status Codes**

Every request will be answered with a proper response and status code. If, for example, a resource has been successfully created, status code 200 (OK) will be returned together with the Id of the newly created resource.

Example: adding a folder to the address book**Request:**

Method	POST /folders
Headers	Content-Type: application/json X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	{"Name":"Map 1","Description":"Some classification"}

**Response:**

On success the following response will be returned:

Status Code	200
Body	857

### 1.3.3 Error messages

CheckMarket's RESTful API returns error messages in your requested format. An error message consist of two attributes:

- Message: contains error-related info;
- Request: the request URI

An example error message in XML could look like this:

```
<Error>
  <Message>Contact with id 12345 does not exist.</Message>
  <Request>/contacts/12345</Request>
</Error>
```

### 1.3.4 List of Status Codes

<u>Code</u>	<u>Name</u>	<u>Description</u>
200	OK	Successful request, correct data will be returned.
400	Bad Request	The request contains a bad syntax and cannot be fulfilled.
401	Unauthorized	Authorization failed. The Api key supplied isn't valid or couldn't be found (you didn't supply the X-CheckMarket-Key header).
404	Not Found	The requested resource couldn't be found.
500	Internal Server Error	Something went wrong on our side. Contact our helpdesk.

### 1.3.5 JSONP

Callback support is currently implemented via JSON with Padding. This way the JSON encoded content will be wrapped in a function call specified via the callback query parameter that should be appended to the query string as follows:

#### **Request:**

Method	GET /folders/0?callback=myFunction
Headers	Content-Type: application/json X-CheckMarket-Key: 5x4f6z84fz67z9z45p

#### **Response:**

On success the following response will be returned:

Status Code	200
Body	myFunction( [{"Id":946,"Name":"Categorie X","Description":"","GroupCount":2,"ContactCount":4}] );

## 1.4 Functions

This section gives an overview of the available functions. Please refer to the Scenarios section for detailed information.

### 1.4.1 Contacts

Method	Resource	Information
GET	contact/{id}	Gets the contact with the specified id.
GET	contacts/{id}	Returns a list of contacts for the given <i>Group</i> id.
POST	contact	Add a single contact to the address book/panel
POST	contacts	Adds multiple contacts to the address book/panel
DELETE	contact/{id}	Deletes the contact with the specified id.

### 1.4.2 Groups

Method	Resource	Information
GET	Groups/{id}	Returns the list of groups in the given folder id. 0 = root
GET	group/{id}	Gets the group with the specified id.
POST	group	Adds a group to the address book.
PUT	group/{id}	Updates the attributes of the specified group.
DELETE	group/{id}	Deletes the group with the specified id. (recursively)

### 1.4.3 Folders

Method	Resource	Information
GET	Folders/{id}	Returns the list of folders in the given folder id. 0 = root
GET	folder/{id}	Gets the folder with the specified id.
POST	folder	Adds a folder to the address book.
PUT	folder/{id}	Updates the attributes of the specified folder.
DELETE	folder/{id}	Deletes the folder with the specified id. (recursively)

### 1.4.4 Surveys

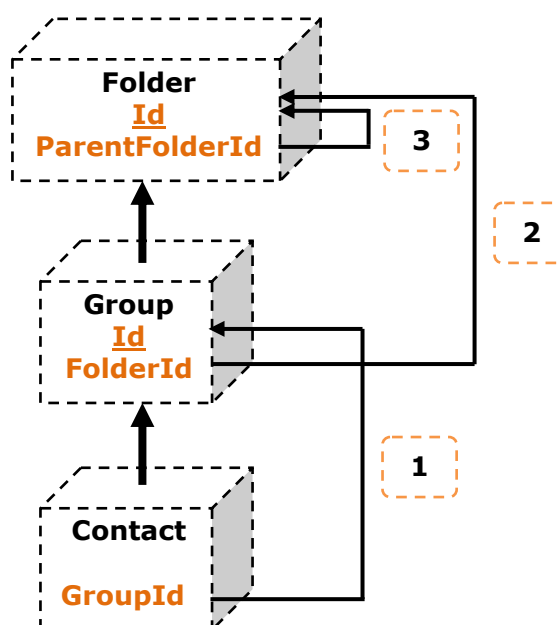
Method	Resource	Information
GET	surveys?lang={language}	language = 2 char lang code .Returns a list of surveys in account.

## 1.5 Scenarios

### 1.5.1 Adding contacts to the address book & panel

It's a good idea to get a grasp on the structure of the address book before diving into different scenarios.

The address book consists of 3 entities that are hierarchically related to each other: Contact, Group, Folder.



Coupling occurs via attributes as shown in the diagram.

- 1) A contact MUST be added to a Group. I.e. the GroupId attribute should reference an existing Group Id. This means that a Group should be present or that a Group should be created before adding a Contact.
- 2) Groups can be nested into Folders via the FolderId of a Group. The FolderId attribute should reference an existing Folder Id.
- 3) Folders can be nested into Folders via the ParentFolderId attribute. ParentFolderId should reference an existing Folder Id.

Note: FolderId and ParentFolderId can be empty. If so, the Group or Folder will be added to the root of the address book.

## 1.5.1.1 Case 1: Adding a single contact, success

**Request:**

Adding a single contact requires the *ContactOptions* and the *Contact* entity to be grouped together in the *AddContact* root element.

Method	POST /contact
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre>&lt;AddContact&gt;   &lt;ContactOptions&gt;     &lt;GroupId&gt;8222&lt;/GroupId&gt;     &lt;SurveyId&gt;1234&lt;/SurveyId&gt;   &lt;/ContactOptions&gt;   &lt;Contact&gt;     &lt;FirstName&gt;Hugo&lt;/FirstName&gt;     &lt;LastName&gt;Leys&lt;/LastName&gt;     &lt;Email&gt;hugo.leys@hotmail.com&lt;/Email&gt;     &lt;LanguageCode&gt;nl&lt;/LanguageCode&gt;     &lt;Street&gt;Van Rotselaerlaan&lt;/Street&gt;     &lt;HouseNumber&gt;374&lt;/HouseNumber&gt;     &lt;PostalCode&gt;2000&lt;/PostalCode&gt;     &lt;City&gt;Antwerpen&lt;/City&gt;     &lt;CountryId&gt;BE&lt;/CountryId&gt;     &lt;Gender&gt;M&lt;/Gender&gt;   &lt;/Contact&gt; &lt;/AddContact&gt;</pre>

**Response:**

On success the Id of the newly added contact will be returned as an integer.

Status Code	200 (Ok)
Body	<int>90687</int>

## 1.5.1.2 Case 2: Adding a single contact, error

**Request:**

There is no GroupId supplied.

Method	POST /contact
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre> &lt;AddContact&gt;    &lt;ContactOptions&gt;     &lt;SurveyId&gt;1234&lt;/SurveyId&gt;   &lt;/ContactOptions&gt;    &lt;Contact&gt;     &lt;FirstName&gt;Hugo&lt;/FirstName&gt;     &lt;LastName&gt;Leys&lt;/LastName&gt;     &lt;Email&gt;hugo.leys@hotmail.com&lt;/Email&gt;     &lt;LanguageCode&gt;nl&lt;/LanguageCode&gt;     &lt;Street&gt;Van Rotselaerlaan&lt;/Street&gt;     &lt;HouseNumber&gt;374&lt;/HouseNumber&gt;     &lt;PostalCode&gt;2000&lt;/PostalCode&gt;     &lt;City&gt;Antwerpen&lt;/City&gt;     &lt;CountryId&gt;BE&lt;/CountryId&gt;     &lt;Gender&gt;M&lt;/Gender&gt;   &lt;/Contact&gt;  &lt;/AddContact&gt; </pre>

**Response:**

A descriptive error message will be returned.

Status Code	404 (Bad Request)
Body	<pre> &lt;Error&gt;   &lt;Message&gt;You must specify a valid GroupId in the ContactOptions entity.&lt;/Message&gt;   &lt;Request&gt;/contact&lt;/Request&gt; &lt;/Error&gt; </pre>

## 1.5.1.3 Case 3: Adding a single contact, error

**Request:**

There is no Email attribute supplied.

Method	POST /contact
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre>&lt;AddContact&gt;    &lt;ContactOptions&gt;     &lt;GroupId&gt;8222&lt;/GroupId&gt;     &lt;SurveyId&gt;1234&lt;/SurveyId&gt;   &lt;/ContactOptions&gt;    &lt;Contact&gt;     &lt;FirstName&gt;Hugo&lt;/FirstName&gt;     &lt;LastName&gt;Leys&lt;/LastName&gt;     &lt;LanguageCode&gt;nl&lt;/LanguageCode&gt;     &lt;Street&gt;Van Rotselaerlaan&lt;/Street&gt;     &lt;HouseNumber&gt;374&lt;/HouseNumber&gt;     &lt;PostalCode&gt;2000&lt;/PostalCode&gt;     &lt;City&gt;Antwerpen&lt;/City&gt;     &lt;CountryId&gt;BE&lt;/CountryId&gt;     &lt;Gender&gt;M&lt;/Gender&gt;   &lt;/Contact&gt;  &lt;/AddContact&gt;</pre>

**Response:**

Since EmailRequired isn't specified its default value (true) will be taken. An error message will be returned.

Status Code	404 (Bad Request)
Body	<pre>&lt;Error&gt;   &lt;Message&gt;The Email attribute is not   valid.&lt;/Message&gt;   &lt;Request&gt;/contact&lt;/Request&gt; &lt;/Error&gt;</pre>

## 1.5.1.4 Case 4: Adding multiple contacts, success

Adding multiple contacts requires the *ContactOptions* and the *Contacts* group-entity to be grouped together in the *AddContacts* root element.

**Request:**

Method	POST /contacts
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre> &lt;AddContacts&gt;    &lt;ContactOptions&gt;     &lt;GroupId&gt;8222&lt;/GroupId&gt;     &lt;SurveyId&gt;1234&lt;/SurveyId&gt;     &lt;DefaultLanguage&gt;nl&lt;/DefaultLanguage&gt;   &lt;/ContactOptions&gt;    &lt;Contacts&gt;     &lt;Contact&gt;       &lt;FirstName&gt;John&lt;/FirstName&gt;       &lt;LastName&gt;Doe&lt;/LastName&gt;       &lt;Email&gt;john.doe@anonymous.com&lt;/Email&gt;     &lt;/Contact&gt;     &lt;Contact&gt;       &lt;FirstName&gt;Jane&lt;/FirstName&gt;       &lt;LastName&gt;Doe&lt;/LastName&gt;       &lt;Email&gt;jane.doe@anonymous.com&lt;/Email&gt;     &lt;/Contact&gt;     &lt;Contact&gt;       &lt;FirstName&gt;Hugo&lt;/FirstName&gt;       &lt;LastName&gt;Leysen&lt;/LastName&gt;       &lt;Email&gt;hugo.leys@hotmail.com&lt;/Email&gt;     &lt;/Contact&gt;     &lt;Contact&gt;       &lt;FirstName&gt;Jan&lt;/FirstName&gt;       &lt;LastName&gt;Smith&lt;/LastName&gt;       &lt;Email&gt;jan@xyz&lt;/Email&gt;     &lt;/Contact&gt;   &lt;/Contacts&gt;  &lt;/AddContacts&gt; </pre>

**Response:**

On success a detailed report will be returned. The report returns the Ids of the successfully added contacts in the *Success* element. The *Failed* element contains a *ContactReport* list which details what went wrong. Both elements display the number of added/failed contacts in the *Amount* element.

Status Code	200 (Ok)
Body	<pre> &lt;Report&gt;   &lt;Success&gt;     &lt;Amount&gt;2&lt;/Amount&gt;     &lt;IdList&gt;       &lt;int&gt;90688&lt;/int&gt;       &lt;int&gt;90689&lt;/int&gt;     &lt;/IdList&gt;   &lt;/Success&gt;    &lt;Failed&gt;     &lt;Amount&gt;2&lt;/Amount&gt;     &lt;FailedList&gt;       &lt;ContactReport&gt;         &lt;Attribute&gt;Email&lt;/Attribute&gt;         &lt;ErrorMessage&gt;A contact with the supplied Email already exists.&lt;/ErrorMessage&gt;         &lt;Contact&gt;           &lt;FirstName&gt;Hugo&lt;/FirstName&gt;           &lt;LastName&gt;Leysen&lt;/LastName&gt;           &lt;Email&gt;hugo.leys@hotmail.com&lt;/Email&gt;         &lt;/Contact&gt;       &lt;/ContactReport&gt;        &lt;ContactReport&gt;         &lt;Attribute&gt;Email&lt;/Attribute&gt;         &lt;ErrorMessage&gt;The Email attribute is not valid.&lt;/ErrorMessage&gt;         &lt;Contact&gt;           &lt;FirstName&gt;Jan&lt;/FirstName&gt;           &lt;LastName&gt;Smith&lt;/LastName&gt;           &lt;Email&gt;jan@xyz&lt;/Email&gt;         &lt;/Contact&gt;       &lt;/ContactReport&gt;     &lt;/FailedList&gt;   &lt;/Failed&gt; &lt;/Report&gt; </pre>

## 1.5.1.5 Case 5: Adding multiple contacts, error

**Request:**

Assuming the supplied SurveyId exists but doesn't have the correct status.

Method	POST /contacts
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre> &lt;AddContacts&gt;    &lt;ContactOptions&gt;     &lt;GroupId&gt;8222&lt;/GroupId&gt;     &lt;SurveyId&gt;54321&lt;/SurveyId&gt;     &lt;EmailRequired&gt;true&lt;/EmailRequired&gt;     &lt;DefaultLanguage&gt;nl&lt;/DefaultLanguage&gt;   &lt;/ContactOptions&gt;    &lt;Contacts&gt;     &lt;Contact&gt;       &lt;FirstName&gt;John&lt;/FirstName&gt;       &lt;LastName&gt;Doe&lt;/LastName&gt;     &lt;/Contact&gt;     &lt;Contact&gt;       &lt;FirstName&gt;Jan&lt;/FirstName&gt;       &lt;LastName&gt;Smith&lt;/LastName&gt;     &lt;/Contact&gt;   &lt;/Contacts&gt;  &lt;/AddContacts&gt; </pre>

**Response:**

The supplied SurveyId hasn't got the correct status assigned. A descriptive error message will be returned.

Status Code	404 (Bad Request)
Body	<pre> &lt;Error&gt;   &lt;Message&gt; Survey with SurveyId '54321' should have "In development" or "Live" status.&lt;/Message&gt;   &lt;Request&gt;/contacts&lt;/Request&gt; &lt;/Error&gt; </pre>

### 1.5.2 Adding a Folder to the address book

A folder is the highest entity in the address book hierarchy. Folders are used to either subdivide Groups to create a logical structure or to nest Folders.

#### 1.5.2.1 Case 1: Adding a folder, success

##### **Request:**

Method	POST /folder
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre>&lt;Folder&gt;   &lt;Name&gt;Map 1&lt;/Name&gt;   &lt;Description&gt;Some classification&lt;/Description&gt; &lt;/Folder&gt;</pre>

##### **Response:**

On success the Id of the folder will be returned.

Status Code	200 (Ok)
Body	<pre>&lt;int&gt;857&lt;/int&gt;</pre>

#### 1.5.2.2 Case 2: Adding a folder, error

##### **Request:**

We'll supply a ParentFolderId that doesn't exist.

Method	POST /folder
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre>&lt;Folder&gt;   &lt;Name&gt;Map 2&lt;/Name&gt;   &lt;Description&gt;Some classification&lt;/Description&gt;   &lt;ParentFolderId&gt;12345&lt;/ParentFolderId&gt; &lt;/Folder&gt;</pre>

##### **Response:**

An appropriate error message will be returned.

Status Code	404 (Bad Request)
Body	<pre>&lt;Error&gt;   &lt;Message&gt;The Parent '12345' does not exist.&lt;/Message&gt; &lt;/Request&gt;/folders&lt;/Request&gt;</pre>

	<code>&lt;/Error&gt;</code>
--	-----------------------------

### 1.5.2.3 Case 3: Updating attributes of a folder

#### **Request:**

We want to nest a folder by updating its ParentFolderId.

Method	PUT /folder/857
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<code>&lt;Folder&gt; &lt;ParentFolderId&gt;4657&lt;/ParentFolderId&gt; &lt;/Folder&gt;</code>

#### **Response:**

On success the appropriate status code is returned.

Status Code	200 (Ok)
-------------	----------

## 1.5.3 Adding a Group to the address book

Groups are used to structure a list of contacts.

### 1.5.3.1 Case 1: Adding a group, success

#### **Request:**

Method	POST /group
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<code>&lt;Group&gt; &lt;Name&gt;Customers&lt;/Name&gt; &lt;Description&gt;Customer group&lt;/Description&gt; &lt;/Group&gt;</code>

#### **Response:**

On success the Id of the Group will be returned.

Status Code	200 (Ok)
Body	<code>&lt;int&gt;256&lt;/int&gt;</code>

## 1.5.3.2 Case 2: Adding a group, error

**Request:**

We'll supply a group name that already exists.

Method	POST /group
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre>&lt;Group&gt;   &lt;Name&gt;Customers&lt;/Name&gt;   &lt;Description&gt;Some group&lt;/Description&gt; &lt;/Group&gt;</pre>

**Response:**

An appropriate error message will be returned.

Status Code	404 (Bad Request)
Body	<pre>&lt;Error&gt;   &lt;Message&gt;A group with name 'Customers'   already exists.&lt;/Message&gt;   &lt;Request&gt;/groups&lt;/Request&gt; &lt;/Error&gt;</pre>

## 1.5.3.3 Case 3: Updating attributes of a group

**Request:**

We want to nest a folder by updating its ParentFolderId.

Method	PUT /group/256
Headers	Content-Type: application/xml X-CheckMarket-Key: 5x4f6z84fz67z9z45p
Body	<pre>&lt;Group&gt;   &lt;FolderId&gt;857&lt;/FolderId&gt; &lt;/Group&gt;</pre>

**Response:**

On success the appropriate status code is returned.

Status Code	200 (Ok)
-------------	----------

## 2 SOAP API

### 2.1 Introduction

This documentation assumes that you are familiar with SOAP and web services. Communication with CheckMarket's SOAP API can be easily established via the WSDL file. The SOAP library in your preferred programming language will auto-generate the operations, data types, etc. The WSDL file can be obtained via URL:

*<https://api.checkmarket.com/SurveyService.svc/mex>*

⚠ Some clients fail to include the correct SOAP endpoint in the WSDL file, please do change the address manually to *<https://api.checkmarket.com/SurveyService.svc/soap>* ⚠

### 2.2 Authentication

Authentication happens via a unique key, the API key, which is connected to your account. The API key can be found in the member section under 'my account'.

Every SOAP call requires the API key to be supplied within a SOAP header named '**CheckMarket-Key**' under the namespace '**api.checkmarket.com**'.

### 2.3 Functions

Note that all functions will return valuable error information via exceptions. Please do use exception handling in your programming language to catch error messages.

#### 2.3.1 Contacts

Function	Parameters	Return type	Information
GetContact	id ( <i>string</i> )	Contact	Gets the contact with the specified id.
GetContactsByGroup	id ( <i>string</i> )	Contact[]	Returns a list of contacts for the given <i>Group</i> id.
AddContact	ContactOptions ( <i>ContactOptions</i> ), Contact ( <i>Contact</i> )	Integer (32 bit)	Add a single contact to the address book/panel
AddContacts	ContactOptions ( <i>ContactOptions</i> ), Contacts ( <i>Contact[]</i> )	Report	Adds multiple contacts to the address book/panel
DeleteContact	id ( <i>string</i> )	Boolean	Deletes the contact with the specified id.

### 2.3.2 Groups

Function	Parameters	Return type	Information
GetGroups	id ( <i>string</i> )	Group[]	Returns the list of groups in the given folder id.
GetGroup	id ( <i>string</i> )	Group	Gets the group with the specified id.
AddGroup	group ( <i>Group</i> )	Integer (32 bit)	Adds a group to the address book.
UpdateGroup	id ( <i>string</i> ), group ( <i>Group</i> )	Boolean	Updates the attributes of the specified group.
DeleteGroup	id ( <i>string</i> )	Boolean	Deletes the group with the specified id. (recursively)

### 2.3.3 Folders

Function	Parameters	Return type	Information
GetFolders	id ( <i>string</i> )	Folder[]	Returns the list of folders in the given folder id.
GetFolder	id ( <i>string</i> )	Folder	Gets the folder with the specified id.
AddFolder	folder ( <i>Folder</i> )	Integer (32 bit)	Adds a folder to the address book.
UpdateFolder	id ( <i>string</i> ), folder ( <i>Folder</i> )	Boolean	Updates the attributes of the specified folder.
DeleteFolder	id ( <i>string</i> )	Boolean	Deletes the folder with the specified id. (recursively)

### 2.3.4 Surveys

Function	Parameters	Return type	Information
GetSurveys	language ( <i>string</i> )	Survey[]	Returns list of surveys in account.
GetSurvey	language ( <i>string</i> )	Survey	Returns the attributes of the specified survey id

## 3 ENTITIES

### 3.1 Contact

Contact in the address book

Resource: contact and contacts

Serialized name: Contact

XML collection: ArrayOfContact

Attributes:

Name	Type	Required	Information
<b>Id</b>	Integer	N/A	Identifier
<b>FirstName</b>	String	No	Given name
<b>LastName</b>	String	No	Family name
<b>Email</b>	String	No	Email address
<b>LanguageCode</b>	String	No	ISO Lang Code
<b>Optional1</b>	String	No	Optional field
<b>Optional2</b>	String	No	Optional field
<b>Optional3</b>	String	No	Optional field
<b>Optional4</b>	String	No	Optional field
<b>Optional5</b>	String	No	Optional field
<b>Street</b>	String	No	Street
<b>HouseNumber</b>	String	No	House number
<b>Suite</b>	String	No	Suite
<b>PostalCode</b>	String	No	Postal code
<b>City</b>	String	No	City
<b>State</b>	String	No	State
<b>Province</b>	String	No	Province
<b>CountryId</b>	String	No	ISO Country Code
<b>Phone</b>	String	No	Phone number
<b>Gender</b>	String	No	M or F

<b>DateOfBirth</b>	String	No	YYYY-MM-DD
<b>Groups</b>	ArrayOfGroup	No	Used to nest contact into a group

### 3.2 ContactOptions

Entity used when adding a single contact or multiple contacts.

Resource: /

Serialized name: ContactOptions

XML collection: ArrayOfContactOptions

Attributes:

Name	Type	Required	Info
<b>GroupId</b>	Integer (32 bit)	Yes	Used when adding contacts
<b>SurveyId</b>	Integer (32 bit)	No	When supplied the contact will be added to the panel of the given survey.
<b>EmailRequired</b>	Boolean	No	true or false, default value is true
<b>EmailUniqueInGroup</b>	Boolean	No	true or false, default value is true
<b>DefaultLanguage</b>	String	No	ISO 2-letter language code

### 3.3 Folder

Folders are used to subdivide groups of contacts in the address book.

Resource: folder and folders

Serialized name: Folder

XML collection: ArrayOfFolder

Attributes:

Name	Type	Required	Info
<b>Id</b>	Integer (16 bit)	N/A	Identifier
<b>Name</b>	String	Yes	Name to be shown in the addressbook
<b>Description</b>	String	No	Optional information to be supplied
<b>ParentFolderId</b>	Integer (16 bit)	No	Used to nest folders in one another.
<b>FolderCount</b>	Integer (16 bit)	N/A	Number of folders in folder and in nested folders.
<b>GroupCount</b>	Integer (16 bit)	N/A	Number of groups in folder and in nested folders.
<b>ContactCount</b>	Integer (16 bit)	N/A	Number if contacts in folder and in nested folders.

### 3.4 Group

Groups are used to subdivide contacts in the address book

Resource: group and groups

Serialized name: Group

XML collection: ArrayOfGroup

Attributes:

Name	Type	Required	Info
<b>Id</b>	Integer (32 bit)	N/A	Identifier
<b>Name</b>	String	Yes	Name to be shown in the addressbook
<b>Description</b>	String	No	Optional information to be supplied
<b>FolderId</b>	Integer (16 bit)	No	Used to nest group into a folder
<b>ContactCount</b>	Integer (16 bit)	N/A	Number of contacts in group

### 3.5 Survey

Representation of a survey

Resource: surveys

Serialized name: Survey

XML collection: ArrayOfSurvey

Attributes:

Name	Type	Required	Info
<b>Id</b>	Integer (32 bit)	N/A	Identifier
<b>Title</b>	Title	N/A	Title of survey
<b>StatusId</b>	Integer (8-bit)	N/A	
<b>StatusString</b>	String	N/A	Status in requested language
<b>CreationDate</b>	String	N/A	Yyyy-mm-dd hh:mm:ss
<b>StartDate</b>	String	N/A	Yyyy-mm-dd hh:mm:ss
<b>EndDate</b>	String	N/A	Yyyy-mm-dd hh:mm:ss
<b>TargetArchiveDate</b>	String	N/A	Yyyy-mm-dd hh:mm:ss
<b>ArchiveDate</b>	String	N/A	Yyyy-mm-dd hh:mm:ss
<b>IsTrial</b>	Boolean	N/A	Indicates if survey is a free trial survey
<b>CreatedBy</b>	String	N/A	Name of user that created survey
<b>PanelCount</b>	Integer (32-bit)	N/A	Number of contacts in panel
<b>RespondentCount</b>	Integer (32-bit)	N/A	Number of respondents
<b>PreviewURL</b>	String	N/A	URL of preview version

### 3.6 Report

Return value when adding multiple contacts

Resource: /

Serialized name: Report

XML collection: /

Attributes:

Name	Type	Required	Info
<b>Success</b>	SuccessReport	N/A	Success report
<b>Failed</b>	FailedReport	N/A	Failed report

### 3.7 SuccessReport

Resource: /

Serialized name: Success

XML collection: /

Attributes:

Name	Type	Required	Info
<b>Amount</b>	Integer (32 bit)	N/A	Amount of successfully added contacts
<b>IdList</b>	Integer	N/A	Ids of added contacts

### 3.8 FailedReport

Resource: /

Serialized name: Failed

XML collection: /

Attributes:

Name	Type	Required	Info
<b>Amount</b>	Integer (32 bit)	N/A	Amount of contacts that were not added
<b>FailedList</b>	ContactReport list	N/A	ContactReports of contacts that were not added

### 3.9 ContactReport

Resource: /

Serialized name: ContactReport

XML collection: FailedList

Attributes:

Name	Type	Required	Info
<b>Attribute</b>	String	N/A	The attribute that caused the error
<b>ErrorMessage</b>	Title	N/A	Description of what went wrong
<b>Contact</b>	Contact	N/A	Contact object