

SmartFocus Cloud Service APIs

Document name	Individual Member Management REST API Guide
Service	Member data management for individual updates and insertions
Protocol	REST
Version	10.12
Last updated on	October 20, 2013

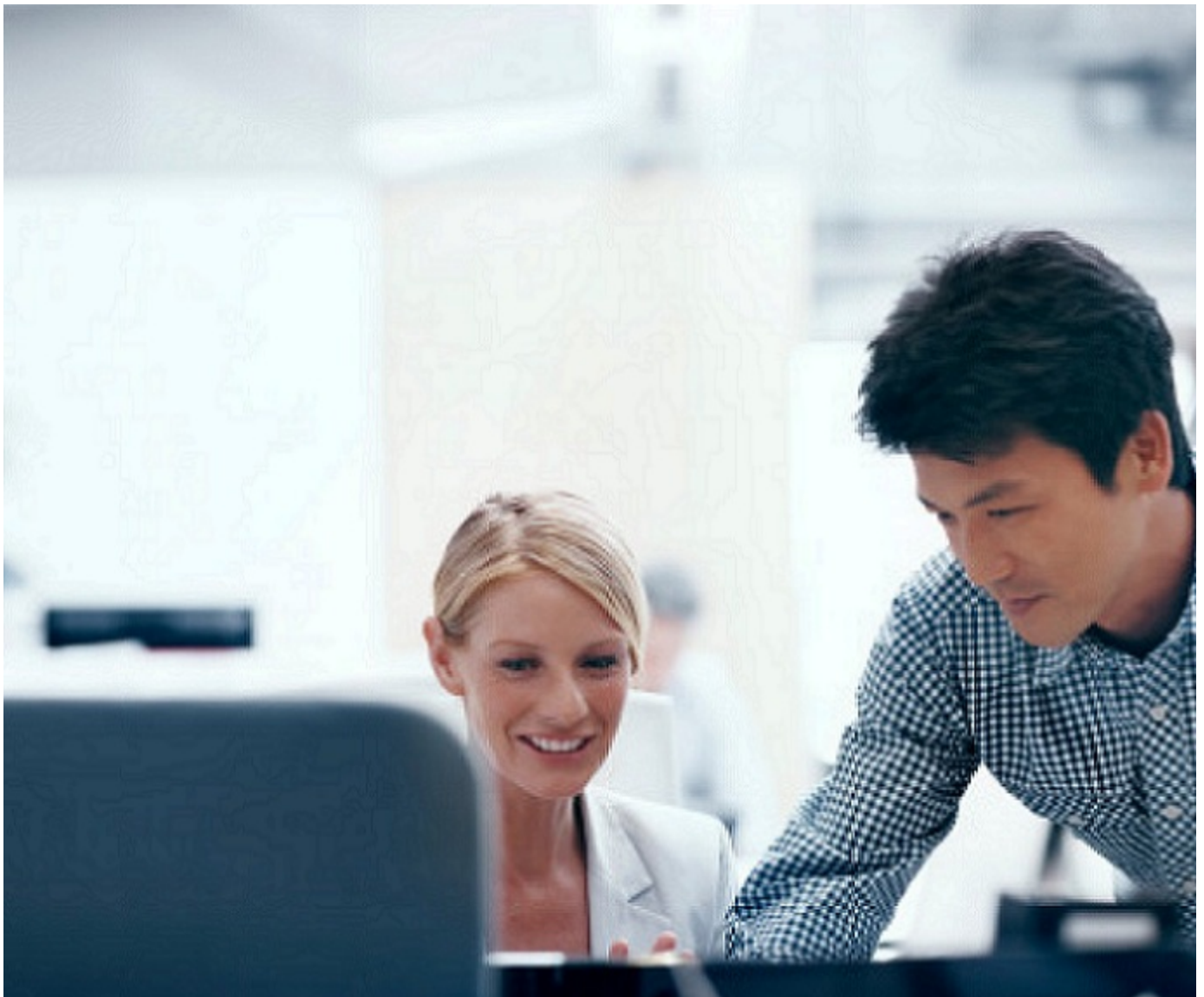


Table of Contents

Table of Contents	2
Introduction	4
About This Document	4
About SmartFocus APIs	4
Feedback	4
Support Options	4
Training Options	4
Useful Links	4
Disclaimer	5
Introducing SmartFocus APIs	6
Module Overview	6
Overview of the Data Individual Update API	7
Data Individual Update API Use Cases	8
Getting Started with Integration	10
Prerequisites	10
Quick Start	10
Integration Using APIs	10
URL Encoding Considerations (for HTTP GET methods only)	12
Security	12
Connection	13
Open Connection	14
Close Connection	16
Insert Member by Email Address	17
REST Example	17
Insert Member (POST)	18
REST Example	18
Update a Single Field (POST)	20
REST Examples	21
Update Member (POST)	22
REST Example	23
Insert or Update Member Data (POST)	24
REST Example	25
Update Member by Email Address	26
REST Example	26
Retrieve the Insert Member Job Status	28
REST Example	28
Retrieve Members by Email Address	30
REST Example	30

Retrieve a Member by Cellphone	34
REST Example	34
Retrieve a Member by ID	38
REST Example	38
Retrieve Members by Page	42
REST Example	42
Retrieve Members by Criteria (POST)	46
REST Example	46
Output	47
Retrieve Member Table Column Names	49
REST Example	49
Unsubscribe a Member by Email Address	52
REST Example	52
Unsubscribe a Member by Cellphone Number	53
REST Example	53
Unsubscribe a Member by ID	54
REST Example	54
Unsubscribe Member by Value	55
REST Examples	55
Re-Subscribe a Member by Email Address	57
REST Example	57
Re-Subscribe a Member by Cellphone Number	58
REST Example	58
Re-Subscribe a Member by ID	60
REST Example	60
Reference	61

Introduction

About This Document

This document is a reference document for using SmartFocus APIs. It does not explain the purpose or functions of SmartFocus features. For information on these features, please consult the *SmartFocus Online Help* or the *SmartFocus User Guide*.

This document is intended for developers and project managers.

About SmartFocus APIs

An Application Programming Interface (API) is a source code interface that a computer system or program library provides in order to support requests for services made from another computer program.

The goal of SmartFocus APIs is to offer customers the ability to pilot a complete campaign from their own system.

Example: The introduction of a new promotional article in the e-commerce back office should automatically generate a new SmartFocus campaign to targeted recipients.

Feedback

The Individual Member Management REST API Guide is constantly being enhanced to provide you with more and more information on using SmartFocus API methods.

If you can't find the information you need or want to provide feedback, simply drop us a line at documentation@smartfocus.com.

We look forward to hearing from you!

Support Options

SmartFocus provides you with a dedicated Account Manager to accompany you throughout the execution of your projects in SmartFocus. Your Account Manager is the gateway to support, training, and professional services. Working with your Account Manager, you can rely on SmartFocus's deliverability and technical support teams for complex troubleshooting and optimization.

Training Options

SmartFocus provides fully comprehensive training ranging from basic product training through to advanced modules and both strategic and tactical marketing courses. The training courses are designed to help you increase productivity, develop new methods, and share best practices to optimize your email marketing campaigns.

The SmartFocus Academy, located in central London and in Paris, has state-of-the-art training rooms and equipment.

To get more information on training, go to the SmartFocus Academy website: <http://www.emailvisionacademy.co.uk>

Useful Links

- Information on SmartFocus's products and services: <http://www.smartfocus.com>
- Training: <http://www.emailvisionacademy.co.uk>

Disclaimer

While the information contained in this publication is believed to be true and accurate, SmartFocus cannot accept any legal responsibility for any errors or omissions contained herein. All information is subject to change without notice.

None of the material in this publication may be reproduced or transmitted in whole or in part without the express written permission of SmartFocus.

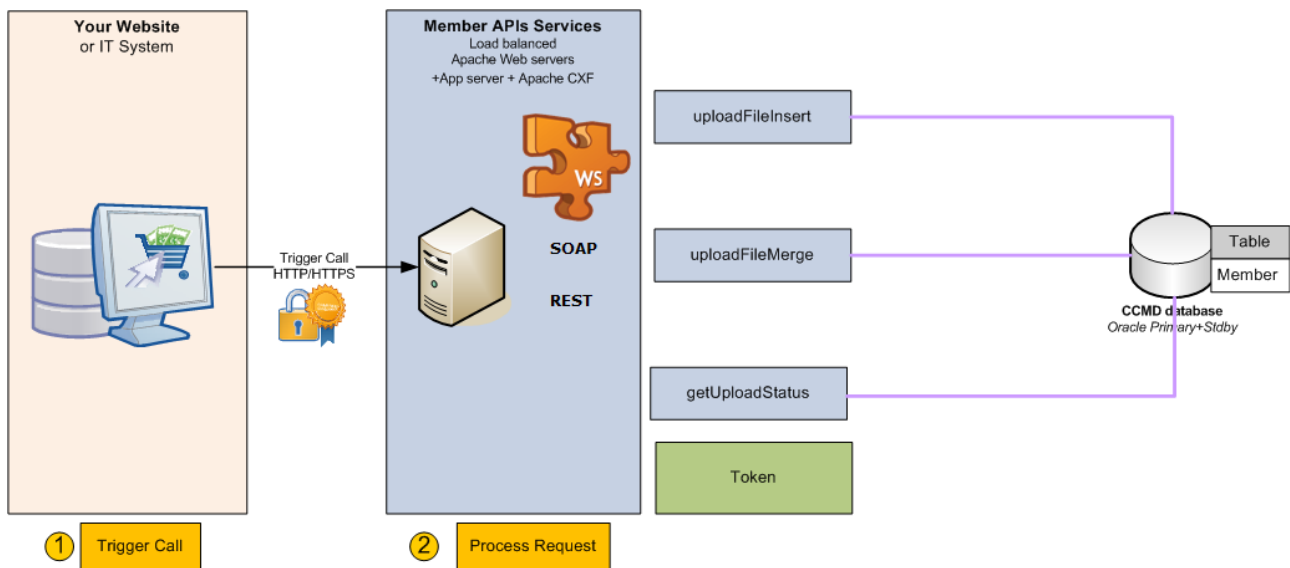
Introducing SmartFocus APIs

Module Overview

The Member module is a way to access your members' profile through the API.

It provides full access to the MEMBER table allowing you to insert, update, and get a member profile.

The member table is a single table stored in SmartFocus's datacenter. It contains all the profile information of your recipients, such as email address, first name, last name, and any column defined during the life of your account.



Overview of the Data Individual Update API

The Data Individual Update API allows you to upload and update individual member's data in your SmartFocus member list.

For further information on uploading and updating member data, please consult the *SmartFocus User Guide* or *SmartFocus Online Help*.

The following methods are available:

Method	Description
Open Connection	This method provides a session token when given valid credentials.
Close Connection	This method terminates the session token.
Insert Member by Email Address	This method inserts a new member using only the email address as input (i.e. all other fields will remain empty).
Insert Member (POST)	This method inserts a new member using the email address as input and providing profile data for other fields of the Member table.
Insert or Update Member Data (POST)	This method searches a specified column of the Member table for a particular value used to identify a member in order to update the member's data. If the member is not found, a new member is created (provided that the email address is given). Any criteria can be used to find the member including one of the fields to be updated.
Update Member by Email Address	This method updates a single field of a member in the Member table using the email address to identify the member. To update another field, another call should be made.
Retrieve the Insert Member Job Status	This method retrieves the job status (i.e. the status of the member insertion or update) using the job ID.
Retrieve Members by Email Address	This method retrieves a list of members who have the given email address.
Retrieve a Member by Cellphone	This method retrieves a list of members who have the given cellphone number.
Retrieve a Member by ID	This method uses the member ID to retrieve the details of a member.
Retrieve Members by Page	This method retrieves all members page by page. Each page contains 10 members.
Retrieve Member Table Column Names	This method retrieves the list of fields (i.e. database column names) available in the Member table.
Unsubscribe a Member by Email Address	This method unsubscribes the member(s) matching the given email address.
Unsubscribe a Member by Cellphone Number	This method unsubscribes one or more members who match a given cellphone number.

Method	Description
Unsubscribe a Member by ID	This method unsubscribes the member matching the given ID.
Re-Subscribe a Member by Email Address	This method re-subscribes an unsubscribed member using his/her email address. If there are multiple members with the same email address, they will all be re-subscribed.
Re-Subscribe a Member by Cellphone Number	Re-subscribes an unsubscribed member using his/her cell phone number. If there are multiple members with the same number, they will all be re-subscribed.
Re-Subscribe a Member by ID	This method re-subscribes the member matching the given ID.
Retrieve Members by Criteria (POST)	This method retrieves a list of members who match the given criteria.
Retrieve Members by Page	This method retrieves all members page by page. Each page contains 10 members.
Update a Single Field (POST)	This method updates a single field of an individual member.
Update Member (POST)	This method updates multiple fields of the member(s). Any criteria can be used to find the member(s) including one of the fields to be updated.
Unsubscribe Member by Value	This method unsubscribes one or more members who match the given criteria.

Data Individual Update API Use Cases

Insert a New Member in the Member List

To upload the member data of a new member into the member table:

1. Use the Open Connection method to open the connection.
2. Use the Insert Member and Member Values method to insert the member data into the member table.
3. Use the Retrieve the Insert Member Job Status method to obtain the status of the insertion.
4. Use the Close Connection method to close the connection.

Unsubscribe a Member

To unsubscribe the member from the mailing list:

1. Use the Open Connection method to open the connection.
2. Use the Retrieve Members by Criteria method to find the member email address of the member you want to update.
3. Use the Unsubscribe a Member by Email Address to unsubscribe the member.

4. Use the Retrieve the Insert Member Job Status method to obtain the status of the unsubscription.
5. Use the Close Connection method to close the connection.

Getting Started with Integration

Prerequisites

To access SmartFocus's APIs and take full advantage of this software's ease of integration with other systems, you will need the following:

- An Internet connection
- A recent browser and operating system
- An active SmartFocus account with the API feature enabled

Quick Start

The process for interfacing your website, CRM, or any other internal system with the APIs is quite straightforward.

Step 1: Get your API key in SmartFocus

Note: You must have a dedicated API login. This login will NOT have access to SmartFocus. Contact your Account Manager to have a dedicated API login.

To connect through the API, the user must first obtain a manager key using the CCMD Web Application.

Calling the connect method (with the login, password, manager key) will provide a token, to be used in all subsequent calls.

This token will expire in the following cases:

- When a close connection call is made.
- When the maximum number of calls per session, defined by the manager in SmartFocus, is reached.
- When the session times out.

Step 2: Build your application

Integration Using APIs

The first step in getting started with web services is to configure the range of remote servers that will access this module.

Webmasters and developers should be able to interface with this new API with any programming language that uses standard HTTP calls.

List of APIs that are available:

- RESTful API
 - HTTP GET QS
 - HTTP GET PI
- SOAP API (see the *Individual Member Management REST API Guide*)

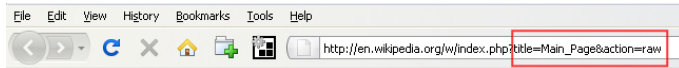
RESTful API

Description: RESTful API is the most standard way to remotely call a web service. REST API requests are always sent over the HTTP protocol and can vary in format and methods of submission.

These API methods are available in two formats: HTTP GET Query String (QS) and HTTP GET Path Info (PI).

- HTTP GET QS (Query String):

- The query string is composed of a series of field-value pairs.
- The field-value pairs are each separated by an equals sign (=).
- The series of pairs is separated by the ampersand (&).
- Below is an Internet browser URL location bar showing a URL where the Query String is: title=Main_page&action=raw



- API call summary:

HTTP GET (Query String)

Submission & sample URL call

`http://{server}/apimember/services/rest/member/getMember?token={token}&email={email}`

Parameters & associated values

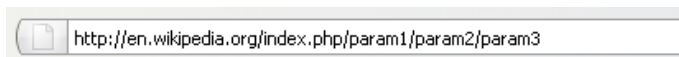
- All parameter names are case-sensitive.
- When specific values are expected, it should be assumed that parameter values are case sensitive.
- The order of parameters must be strictly followed.

- HTTP GET PI (Path Info):

The result is identical to the Query String method. It differs in the way parameters are organized in the URL. In the PI method, parameters are organized like a path.

The order of all parameters is very important.

- The path is composed of a series of values.
- The values are each separated by a forward slash sign (/).
- Below is an Internet browser URL location bar showing a URL with the Path Info:



- API call summary:

HTTP GET (Path Info)

Submission URL

`http://{server}/apimember/services/rest/member/getMember/{token}/{email}`

Parameters & associated values

- All parameter names are case-sensitive.
- The order of parameters must be strictly followed.

Example call

`http://{server}/apimember/services/rest/member/getMember/{token}
/johnsmith@smartfocus.com`

URL Encoding Considerations (for HTTP GET methods only)

Some characters cannot be part of a URL - for example, spaces are not allowed. Some characters have a special meaning in a URL—for example, the hash (#) character is used to locate a specific point within a page, and the equals (=) character is used to separate a name from a value. A query string may need to be converted to satisfy these constraints. This can be done using a schema known as URL encoding. In particular, encoding the query string uses the following rules:

- Letters (A-Z and a-z) and numbers (0-9) are not encoded.
- The period (.), comma (,) , tilde (~), and underscore (_) characters are not encoded.
- A space is encoded as %20.
- The forward slash (/) is encoded as %2F.
- All other characters are encoded as %FF hex representation with any non-ASCII characters first encoded as UTF-8 (or other specified encoding).
- To encode as RFC 1738, use the + sign to replace spaces.

Security

As web services are accessible over the Internet and can be interfaced with any system, there is a risk of fraudulent access and usage of the system. To tighten the security, SmartFocus APIs can be accessed using the HTTPS protocol coupled with the use of encrypted and matching tags.

To use HTTPS, just replace **HTTP** with **HTTPS** in all the submission URLs.

Connection

Prerequisite: To use SmartFocus APIs, you need to have the API manager login provided by SmartFocus and the associated password.

To connect through the API, you must first retrieve the manager key from SmartFocus.

1. Go to **Account Administration** and select **Logins**.
2. Click the **Edit** icon next to your API manager.
3. In the **API** section of the popup window, copy the API key (also known as the manager key) and use it to open a connection to retrieve the token that will be used in your calls.

Calling the connect method (with the login, password, manager key) will provide a token, to be used in all subsequent calls.

This token will expire in the following cases:

- When a close connection call is made.
- When the maximum number of calls per session, defined by the manager in SmartFocus, is reached.
- When the session times out.

Open Connection

This method provides a session token when given valid credentials.

Note: The token is valid for 60 minutes. To avoid problems with expired tokens, it is recommended that you close your connection after an API call and open a new connection for a new API call.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/connect/open?login={apiLogin}&password={apiPassword}&key={apiKey}`

REST PI syntax:

`https://{server}/apimember/services/rest/connect/open/{apiLogin}/{apiPassword}/{apiKey}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
login	The login provided for API access	return	The token to use in all other API calls
pwd	The password Note: API passwords expire after 365 days.		
key	The manager key copied from SmartFocus (see Connection on page 13)		

Error messages

You must fill in the apiName parameter to check rights of client on this API.
You must fill in the login parameter to authenticate on this API.
You must fill in the password parameter to authenticate on this API.
You must fill in the managerKey parameter to authenticate on this API.
Error while decoding managerKey.
Your login is not valid !!
Your password is not valid !!
No manager retrieved for those login, password.
No available connection for manager {0}.
{0} doesn't exist or is not activated on client account.
{0} is not activated for the client.

Error messages

This manager does not have authorized access to this API.

Error while parsing validDate on managerKey.

Date not valid on managerKey!

The managerKey is no longer valid. Your API access is closed!

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/connect/open?login=John&password=afj23rlaf&key=CdX7Cr1E_26blFNJOsgfdawh6LJ3y6pwg5PEOvA`

REST PI

`https://{server}/apimember/services/rest/connect/open/John/afj23rlaf/CdX7Cr1E_26blFNJOsgfdawh6LJ3y6pwg5PEOvA`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:string">
    {token}
  </result>
</response>
```

Close Connection

This method terminates the session token.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/connect/close?token={token}`

REST PI syntax:

`https://{server}/apimember/services/rest/connect/close/{token}`

Note: Ask your Account Manager for your server name.

Input parameter	Description	Output parameter	Description
Required parameters			
token	The connection token	return	The connection is closed if the operation was successful, otherwise an error code appears.

Error messages

You must fill in the token parameter

No available connection for the specified token.

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/connect/close?token={token}`

REST PI

`https://{server}/apimember/services/rest/connect/close/{token}`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:string">connection closed</result>
</response>
```


Insert Member by Email Address

This method inserts a new member using only the email address as input (i.e. all other fields will remain empty).

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/insert?token={token}&email={emailAddress}`

Rest PI syntax:

`https://{server}/apimember/services/rest/member/insert/{token}/{emailAddress}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the insertion
email	The email address		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/insert?token={token}&email=johnsmith@smartfocus.com`

REST PI

`https://{server}/apimember/services/rest/member/insert/{token}/johnsmith@smartfocus.com`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Insert Member (POST)

This method inserts a new member using the email address as input and providing profile data for other fields of the Member table.

This is a POST method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

`https://{server}/apimember/services/rest/member/insertMember/{token}`

Note: Ask your Account Manager for your server name.

Input parameters Required parameters	Description	Output parameters	Description
token	The connection token	return	The job ID of the insertion
email	The email address		
dynContent	The dynContent envelope containing the list of fields to be updated and their values.		
entry	The entry envelope containing the field to update and its value.		
key	The field that will be updated.		
value	The value with which to update the field. Note: For number fields accepting decimal values, you must use the period (.) as the decimal separator. This only applies to fields that are configured to accept decimal values.		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

URL

`https://{server}/apimember/services/rest/member/insertMember/{token}`

HTTP Method

POST

Header

`content-type application/xml`

`accept application/xml`

Body

```
<?xml version="1.0" encoding="utf-8"?>
  <synchroMember>
    <dynContent>
      <entry>
        <key>FIRSTNAME</key>
        <value>John</value>
      </entry>
      <entry>
        <key>LASTNAME</key>
        <value>Smith</value>
      </entry>
    </dynContent>
    <email>johnsmith@example.com</email>
  </synchroMember>
```

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Update a Single Field (POST)

This method updates a single field of an individual member.

Note: It should be noted that if the criteria used is the same for multiple members, all of these members will be updated.

Note: Multiple sets of criteria can be combined to identify the member, e.g.
FIRSTNAME:John|LASTNAME:Smith|EMAIL:johnsmith@smartfocus.com

This is a POST method.

REST WADL

https://{server}/apimember/services/rest?_wadl&_type=xml

REST URL

<https://{server}/apimember/services/rest/member/updateMember/{token}>

Note: Ask your Account Manager for your server name.

Input parameters Required parameters	Description	Output parameters	Description
token	The connection token	return	The job ID of the insertion or update
memberUID	The field or fields used to identify the member and their values in the following format: <ul style="list-style-type: none">Field1:Value1 Field2:Value2		
dynContent	The dynContent envelope containing the list of fields to be updated and their values.		
entry	The entry envelope containing the field to update and its value.		
key	The field that will be updated.		
value	The value with which to update the field. Note: For number fields accepting decimal values, you must use the period (.) as the decimal separator. This only applies to fields that are configured to accept decimal values.		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Examples

Example 1

Input

```
https://{server}/apimember/services/rest/member/updateMember/{token}
<?xml version="1.0" encoding="UTF-8"?>
  <synchroMember>
    <memberUID>{field1}:{value1}</memberUID>
    <dynContent>
      <entry>
        <key>FIRSTNAME</key>
        <value>{newValue}</value>
      </entry>
    </dynContent>
  </synchroMember>
```

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Example 2

Example with Multiple Sets of Criteria

Input

```
https://{server}/apimember/services/rest/member/updateMember/{token}<?xml version="1.0" encoding="UTF-8"?>
  <synchroMember>
    <memberUID>{field1}:{value1}||{field2}:{value2}</memberUID>
    <dynContent>
      <entry>
        <key>FIRSTNAME</key>
        <value>{newValue}</value>
      </entry>
    </dynContent>
  </synchroMember>
```

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Update Member (POST)

This method updates multiple fields of the member(s). Any criteria can be used to find the member(s) including one of the fields to be updated.

The **memberUID** attribute is used to specify the **key** and **value** used as search criteria. The **dynContent** attribute should only contain the values to be updated.

Note: It should be noted that if the criteria used is the same for multiple members, all of these members will be updated.

Note: Multiple sets of criteria can be combined to identify the member, e.g.
FIRSTNAME:John|LASTNAME:Smith|EMAIL:johnsmith@smartfocus.com

This is a POST method.

REST WADL

https://{server}/apimember/services/rest?_wadl&_type=xml

REST URL

<https://{server}/apimember/services/rest/member/updateMember/{token}>

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the update
memberUID	The field or fields used to identify the member and their values in the following format: <ul style="list-style-type: none"> Field1:Value1 Field2:Value2 		
dynContent	The dynContent envelope containing the list of fields to be updated and their values.		
entry	The entry envelope containing the field to update and its value.		
key	The field that will be updated.		
value	The value with which to update the field. Note: For number fields accepting decimal values, you must use the period (.) as the decimal separator. This only applies to fields that are configured to accept decimal values.		

Error messages

You must fill in the token parameter
An error occurred on the server

REST Example

Input

URL

`https://{server}/apimember/services/rest/member/updateMember/{token}`

HTTP Method

POST

Header

`content-type application/xml`

`accept application/xml`

Body

```
<?xml version="1.0" encoding="utf-8"?>
  <synchroMember>
    <memberUID>EMAIL:johnsmith@example.com</memberUID>
  <dynContent>
    <entry>
      <key>EMAIL</key>
      <value>johnsmith@example.com</value>
    </entry>
    <entry>
      <key>FIRSTNAME</key>
      <value>Johnny</value>
    </entry>
  </dynContent>
</synchroMember>
```

Insert or Update Member Data (POST)

This method searches a specified column of the Member table for a particular value used to identify a member in order to update the member's data. If the member is not found, a new member is created (provided that the email address is given). Any criteria can be used to find the member including one of the fields to be updated.

The **memberUID** attribute is used to specify the **key** and **value** used as search criteria. The **dynContent** attribute should only contain the values to be updated.

Note: It should be noted that if the criteria used is the same for multiple members, all of these members will be updated.

Note: Multiple sets of criteria can be combined to identify the member, e.g.
FIRSTNAME:John|LASTNAME:Smith|EMAIL:johnsmith@smartfocus.com

This is a POST method.

REST WADL

https://{server}/apimember/services/rest?_wadl&_type=xml

REST URL

<https://{server}/apimember/services/rest/member/insertOrUpdateMember/{token}>

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the insertion or update
memberUID	The field or fields used to identify the member and their values in the following format: <ul style="list-style-type: none"> Field1:Value1 Field2:Value2 		
dynContent	The dynContent envelope containing the list of fields to be updated and their values.		
entry	The entry envelope containing the field to update and its value.		
key	The field that will be updated.		
value	The value with which to update the field. Note: For number fields accepting decimal values, you must use the period (.) as the decimal separator. This only applies to fields that are configured to accept decimal values.		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

URL

`https://{server}/apimember/services/rest/member/insertOrUpdateMember/{token}`

HTTP Method

POST

Header

`content-type application/xml`

`accept application/xml`

Body

```
<?xml version="1.0" encoding="utf-8"?>
<synchroMember>
  <memberUID>{field1}:{value1}|{field2}:{value2}</memberUID>
  <dynContent>
    <entry>
      <key>{fieldNameA}</key>
      <value>{fieldValueA}</value>
    </entry>
    <entry>
      <key>{fieldNameB}</key>
      <value>{fieldValueB}</value>
    </entry>
    <entry>
      <key>{fieldNameC}</key>
      <value>{fieldValueC}</value>
    </entry>
  </dynContent>
</synchroMember>
```

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Update Member by Email Address

This method updates a single field of a member in the Member table using the email address to identify the member. To update another field, another call should be made.

Note: If multiple members share the same email address, the field will be updated for all members.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/update?token={token}&email={emailAddress}&field={fieldName}&value={fieldValue}`

Rest PI syntax:

`https://{server}/apimember/services/rest/member/update/{token}/{emailAddress}/{fieldName}/{fieldValue}`

Note: Ask your Account Manager for your server name.

Input parameters Required parameters	Description	Output parameters	Description
token	The connection token	return	The job ID of the update
email	The email address		
field	The field that will be updated.		
value	The value with which to update the field. <div>Note: For number fields accepting decimal values, you must use the period (.) as the decimal separator. This only applies to fields that are configured to accept decimal values.</div>		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/update?token={token}&email=l=johnsmith@example.com
&field=FIRSTNAME&value=John`

REST PI

`https://{server}/apimember/services/rest/member/update/{token}/johnsmith@example.com/FIRSTNAME/John`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370852</result>
</response>
```

Retrieve the Insert Member Job Status

This method retrieves the job status (i.e. the status of the member insertion or update) using the job ID.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/getMemberJobStatus?token={token}&synchroId={synchroId}`

Rest PI syntax:

`https://{server}/apimember/services/rest/member/getMemberJobStatus/{token}/{synchroId}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The email address of the member
synchroid	The job ID		The job status: <ul style="list-style-type: none">• Insert• Processing• Processed• Error• Job_Done_Or_Does_Not_Exist

Error messages

You must fill in the token parameter

You must fill in the synchroid parameter.

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/getMemberJobStatus?token={token}&synchroId=378`

REST PI

`https://{server}/apimember/services/rest/member/getMemberJobStatus/{token}/378`

Output

```
<response responseStatus="success">
  <processStatus>
    <email>johnsmith@example.com</email>
```

```
<status>Insert</status>  
</processStatus>  
</response>
```

Retrieve Members by Email Address

This method retrieves a list of members who have the given email address.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/getMemberByEmail?token={token}&email={emailAddress}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/getMemberByEmail/{token}/{emailAddress}`

Note: Ask your Account Manager for your server name.

Input parameters Required parameters	Description	Output parameters	Description
token	The connection token	return	The list of values in the Member table for the member(s)
Email (key)	The email address of the member(s) to retrieve		

Error messages

You must fill in the token parameter

You must fill in the email parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/getMemberByEmail?token={token}&email=l=johnsmith@smartfocus.com`

REST PI

`https://{server}/apimember/services/rest/member/getMemberByEmail/{token}/johnsmith@smartfocus.com`

Output

```
<response responseStatus="success">
  <members>
    <member>
      <attributes>
        <entry>
          <key>SMS_DATE_UNJOIN</key>
```

```
</entry>
<entry>
  <key>EMVADMIN5</key>
</entry>
<entry>
  <key>EMVADMIN4</key>
</entry>
<entry>
  <key>EMVADMIN3</key>
</entry>
<entry>
  <key>EMVADMIN2</key>
</entry>
<entry>
  <key>MEMBER_ID</key>
  <value xsi:type="xs:long">11046926</value>
</entry>
<entry>
  <key>EMVADMIN1</key>
</entry>
<entry>
  <key>FIRSTNAME</key>
  <value xsi:type="xs:string">John</value>
</entry>
<entry>
  <key>EMVCELLPHONE</key>
</entry>
<entry>
  <key>DATEUNJOIN</key>
</entry>
<entry>
  <key>SOURCE</key>
</entry>
<entry>
  <key>SEED</key>
</entry>
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">johnsmith@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2011-02-08T15:34:13+01:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
```

```
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">P</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
<member>
  <attributes>
    <entry>
      <key>SMS_DATE_UNJOIN</key>
    </entry>
    <entry>
      <key>EMVADMIN5</key>
    </entry>
    <entry>
      <key>EMVADMIN4</key>
    </entry>
    <entry>
      <key>EMVADMIN3</key>
    </entry>
    <entry>
      <key>EMVADMIN2</key>
    </entry>
    <entry>
      <key>MEMBER_ID</key>
      <value xsi:type="xs:long">200534</value>
    </entry>
    <entry>
      <key>EMVADMIN1</key>
    </entry>
    <entry>
      <key>FIRSTNAME</key>
      <value xsi:type="xs:string">John</value>
    </entry>
    <entry>
      <key>EMVCELLPHONE</key>
      <value xsi:type="xs:long">61234567</value>
    </entry>
    <entry>
      <key>DATEUNJOIN</key>
    </entry>
    <entry>
      <key>SOURCE</key>
    </entry>
  </attributes>
</member>
```



```
<key>SEED</key>
</entry>
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">johnsmith@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2010-09-24T10:16:46+02:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">Smith</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
</members>
</response>
```

Retrieve a Member by Cellphone

This method retrieves a list of members who have the given cellphone number.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/getMemberByCellphone?token={token}&cellphone={cellphone}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/getMemberByCellphone/{token}/{cellphone}`

Note: Ask your Account Manager for your server name.

Input parameters Required parameters	Description	Output parameters	Description
token	The connection token	return	The list of values in the Member table for the member(s)
cellphone	The cellphone number of the member		

Error messages

You must fill in the token parameter

You must fill in the cellphone parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/getMemberByCellphone?token={token}&cellphone=061234567`

REST PI

`https://{server}/apimember/services/rest/member/getMemberByCellphone/{token}/061234567`

Output

```
<response responseStatus="success">
  <members>
    <member>
```

```
<attributes>
  <entry>
    <key>SMS_DATE_UNJOIN</key>
  </entry>
  <entry>
    <key>EMVADMIN5</key>
  </entry>
  <entry>
    <key>EMVADMIN4</key>
  </entry>
  <entry>
    <key>EMVADMIN3</key>
  </entry>
  <entry>
    <key>EMVADMIN2</key>
  </entry>
  <entry>
    <key>MEMBER_ID</key>
    <value xsi:type="xs:long">11046926</value>
  </entry>
  <entry>
    <key>EMVADMIN1</key>
  </entry>
  <entry>
    <key>FIRSTNAME</key>
    <value xsi:type="xs:string">John</value>
  </entry>
  <entry>
    <key>EMVCELLPHONE</key>
  </entry>
  <entry>
    <key>DATEUNJOIN</key>
  </entry>
  <entry>
    <key>SOURCE</key>
  </entry>
  <entry>
    <key>SEED</key>
  </entry>
  <entry>
    <key>EMVHBQ</key>
  </entry>
  <entry>
    <key>EMAIL</key>
    <value xsi:type="xs:string">johnsmith@example.com</value>
  </entry>
  <entry>
    <key>CODE</key>
  </entry>
  <entry>
    <key>EMVUNROUTABLE</key>
  </entry>
  <entry>
    <key>EMVISP</key>
  </entry>
  <entry>
    <key>LHE</key>
  </entry>
  <entry>
    <key>DATEOFBIRTH</key>
  </entry>
  <entry>
    <key>DATEJOIN</key>
    <value xsi:type="xs:dateTime">2011-02-08T15:34:13+01:00</value>
  </entry>
</attributes>
```

```
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">P</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
<member>
  <attributes>
    <entry>
      <key>SMS_DATE_UNJOIN</key>
    </entry>
    <entry>
      <key>EMVADMIN5</key>
    </entry>
    <entry>
      <key>EMVADMIN4</key>
    </entry>
    <entry>
      <key>EMVADMIN3</key>
    </entry>
    <entry>
      <key>EMVADMIN2</key>
    </entry>
    <entry>
      <key>MEMBER_ID</key>
      <value xsi:type="xs:long">200534</value>
    </entry>
    <entry>
      <key>EMVADMIN1</key>
    </entry>
    <entry>
      <key>FIRSTNAME</key>
      <value xsi:type="xs:string">John</value>
    </entry>
    <entry>
      <key>EMVCELLPHONE</key>
      <value xsi:type="xs:long">61234567</value>
    </entry>
    <entry>
      <key>DATEUNJOIN</key>
    </entry>
  </attributes>
</member>
```

```
<key>SOURCE</key>
</entry>
<entry>
  <key>SEED</key>
</entry>
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">johnsmith@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2010-09-24T10:16:46+02:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">Smith</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
</members>
</response>
```

Retrieve a Member by ID

This method uses the member ID to retrieve the details of a member.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/getMemberById?token={token}&id={memberId}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/getMemberById/{token}/{memberId}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The list of values in the Member table for the member
Id (key)	The ID of the member whose details you want to retrieve		

Error messages

You must fill in the token parameter

You must fill in the id parameter.

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/getMemberById?token={token}&id=385478`

REST PI

`https://{server}/apimember/services/rest/member/getMemberById/{token}/385478`

Output

```
<response responseStatus="success">
  <members>
    <member>
      <attributes>
        <entry>
          <key>SMS_DATE_UNJOIN</key>
        </entry>
      </attributes>
    </member>
  </members>
</response>
```

```
<entry>
  <key>EMVADMIN5</key>
</entry>
<entry>
  <key>EMVADMIN4</key>
</entry>
<entry>
  <key>EMVADMIN3</key>
</entry>
<entry>
  <key>EMVADMIN2</key>
</entry>
<entry>
  <key>MEMBER_ID</key>
  <value xsi:type="xs:long">11046926</value>
</entry>
<entry>
  <key>EMVADMIN1</key>
</entry>
<entry>
  <key>FIRSTNAME</key>
  <value xsi:type="xs:string">John</value>
</entry>
<entry>
  <key>EMVCELLPHONE</key>
</entry>
<entry>
  <key>DATEUNJOIN</key>
</entry>
<entry>
  <key>SOURCE</key>
</entry>
<entry>
  <key>SEED</key>
</entry>
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">johnsmith@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2011-02-08T15:34:13+01:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
```

```
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">P</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
<member>
  <attributes>
    <entry>
      <key>SMS_DATE_UNJOIN</key>
    </entry>
    <entry>
      <key>EMVADMIN5</key>
    </entry>
    <entry>
      <key>EMVADMIN4</key>
    </entry>
    <entry>
      <key>EMVADMIN3</key>
    </entry>
    <entry>
      <key>EMVADMIN2</key>
    </entry>
    <entry>
      <key>MEMBER_ID</key>
      <value xsi:type="xs:long">200534</value>
    </entry>
    <entry>
      <key>EMVADMIN1</key>
    </entry>
    <entry>
      <key>FIRSTNAME</key>
      <value xsi:type="xs:string">John</value>
    </entry>
    <entry>
      <key>EMVCELLPHONE</key>
      <value xsi:type="xs:long">61234567</value>
    </entry>
    <entry>
      <key>DATEUNJOIN</key>
    </entry>
    <entry>
      <key>SOURCE</key>
    </entry>
    <entry>
      <key>SEED</key>
    </entry>
  </attributes>
</member>
```



```
</entry>
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">johnsmith@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2010-09-24T10:16:46+02:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">Smith</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
</members>
</response>
```

Retrieve Members by Page

This method retrieves all members page by page. Each page contains 10 members.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/getListMembersByPage?token={token}&page={page}`

REST PI syntax:

`https://{server}/apimember/services/rest/getListMembersByPage/{token}/{page}`

Note: Ask your Account Manager for your server name.

Input parameters Required parameters	Description	Output parameters	Description
token	The connection token	return	The page object containing the list of members
page	The page number to retrieve		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/getListMembersByPage?token={token}&page=1`

REST PI

`https://{server}/apimember/services/rest/getListMembersByPage/{token}/1`

Output

```
<response responseStatus="success">
  <result xsi:type="apiPagination">
    <currentPage>1</currentPage>
    <list xsi:type="apiMember">
      <attributes>
        <entry>
          <key>SMS_DATE_UNJOIN</key>
        </entry>
        <entry>
          <key>EMVADMIN5</key>
        </entry>
      </attributes>
    </list>
  </result>
</response>
```

```
</entry>
<entry>
  <key>EMVADMIN4</key>
</entry>
<entry>
  <key>EMVADMIN3</key>
</entry>
<entry>
  <key>EMVADMIN2</key>
</entry>
<entry>
  <key>MEMBER_ID</key>
  <value xsi:type="xs:long">39246482</value>
</entry>
<entry>
  <key>EMVADMIN1</key>
</entry>
<entry>
  <key>FIRSTNAME</key>
  <value xsi:type="xs:string">Jane</value>
</entry>
<entry>
  <key>EMVCELLPHONE</key>
</entry>
<entry>
  <key>DATEUNJOIN</key>
</entry>
<entry>
  <key>SOURCE</key>
  <value xsi:type="xs:string">loyalty</value>
</entry>
<entry>
  <key>SEED</key>
</entry>
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">janescott@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2012-04-10T03:51:01+02:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
```

```
<key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">Scott</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</list>
<list xsi:type="apiMember">
  <attributes>
    <entry>
      <key>SMS_DATE_UNJOIN</key>
    </entry>
    <entry>
      <key>EMVADMIN5</key>
    </entry>
    <entry>
      <key>EMVADMIN4</key>
    </entry>
    <entry>
      <key>EMVADMIN3</key>
    </entry>
    <entry>
      <key>EMVADMIN2</key>
    </entry>
    <entry>
      <key>MEMBER_ID</key>
      <value xsi:type="xs:long">39246481</value>
    </entry>
    <entry>
      <key>EMVADMIN1</key>
    </entry>
    <entry>
      <key>FIRSTNAME</key>
      <value xsi:type="xs:string">John</value>
    </entry>
    <entry>
      <key>EMVCELLPHONE</key>
    </entry>
    <entry>
      <key>DATEUNJOIN</key>
    </entry>
    <entry>
      <key>SOURCE</key>
      <value xsi:type="xs:string">loyalty</value>
    </entry>
    <entry>
      <key>SEED</key>
    </entry>
```

```
<entry>
  <key>EMVHBQ</key>
</entry>
<entry>
  <key>EMAIL</key>
  <value xsi:type="xs:string">johnsmith@example.com</value>
</entry>
<entry>
  <key>CODE</key>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
</entry>
<entry>
  <key>EMVISP</key>
</entry>
<entry>
  <key>LHE</key>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2012-04-10T03:51:01+02:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">Smith</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</list>
<nbItemsPerPage>10</nbItemsPerPage>
<nbPages>62</nbPages>
<nbTotalItems>616</nbTotalItems>
<nextPage>true</nextPage>
</result>
</response>
```

Retrieve Members by Criteria (POST)

This method retrieves a list of members who match the given criteria.

This is a POST method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

`https://{server}/apimember/services/rest/member/getMembers/{token}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the insertion or update
memberUID	The field or fields used to identify the member and their values in the following format: <ul style="list-style-type: none">Field1:Value1 Field2:Value2		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

URL

`https://{server}/apimember/services/rest/member/getMembers/{token}`

HTTP Method

POST

Header

`content-type application/xml`

`accept application/xml`

Body

```
<synchroMember>
  <memberUID>FIRSTNAME:John|LASTNAME:Smith</memberUID>
</synchroMember>
```

Output

```
<response responseStatus="success">
  <members>
    <member>
      <attributes>
        <entry>
          <key>SMS_DATE_UNJOIN</key>
        </entry>
        <entry>
          <key>EMVADMIN5</key>
        </entry>
        <entry>
          <key>EMVADMIN4</key>
        </entry>
        <entry>
          <key>EMVADMIN3</key>
        </entry>
        <entry>
          <key>EMVADMIN2</key>
        </entry>
        <entry>
          <key>MEMBER_ID</key>
          <value xsi:type="xs:long">321566</value>
        </entry>
        <entry>
          <key>EMVADMIN1</key>
        </entry>
        <entry>
          <key>FIRSTNAME</key>
          <value xsi:type="xs:string">John</value>
        </entry>
        <entry>
          <key>EMVCELLPHONE</key>
          <value xsi:type="xs:long">61234567</value>
        </entry>
        <entry>
          <key>DATEUNJOIN</key>
        </entry>
        <entry>
          <key>SOURCE</key>
        </entry>
        <entry>
          <key>SEED</key>
        </entry>
        <entry>
          <key>EMVHBQ</key>
        </entry>
        <entry>
          <key>EMAIL</key>
          <value xsi:type="xs:string">johnsmith@example.com</value>
        </entry>
        <entry>
          <key>CODE</key>
        </entry>
        <entry>
          <key>EMVUNROUTABLE</key>
        </entry>
        <entry>
          <key>EMVISP</key>
        </entry>
        <entry>
          <key>LHE</key>
        </entry>
      </attributes>
    </member>
  </members>
</response>
```

```
</entry>
<entry>
  <key>DATEOFBIRTH</key>
</entry>
<entry>
  <key>DATEJOIN</key>
  <value xsi:type="xs:dateTime">2010-10-20T15:49:10+02:00</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
</entry>
<entry>
  <key>HBQ_REASON</key>
</entry>
<entry>
  <key>CLIENTURN</key>
</entry>
<entry>
  <key>SEGMENT</key>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
</entry>
<entry>
  <key>LASTNAME</key>
  <value xsi:type="xs:string">Smith</value>
</entry>
<entry>
  <key>TITLE</key>
</entry>
</attributes>
</member>
</members>
</response>
```


Retrieve Member Table Column Names

This method retrieves the list of fields (i.e. database column names) available in the Member table.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/descMemberTable?token={token}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/descMemberTable/{token}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The list of database column names

Error messages

You must fill in the token parameter
An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/descMemberTable?token={token}`

REST PI

`https://{server}/apimember/services/rest/member/descMemberTable/{token}`

Output

```
<response responseStatus="success">
  <memberTable>
    <fields>
      <entry>
        <key>CLIENTURN</key>
        <value>STRING</value>
      </entry>
      <entry>
        <key>CLIENT_ID</key>
        <value>LONG</value>
      </entry>
      <entry>
        <key>CODE</key>
        <value>LONG</value>
      </entry>
    </fields>
  </memberTable>
</response>
```

```
<entry>
  <key>DATEJOIN</key>
  <value>DATE</value>
</entry>
<entry>
  <key>DATEOFBIRTH</key>
  <value>DATE</value>
</entry>
<entry>
  <key>DATEUNJOIN</key>
  <value>DATE</value>
</entry>
<entry>
  <key>EMAIL</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMAIL_FORMAT</key>
  <value>LONG</value>
</entry>
<entry>
  <key>EMAIL_ORIGINE</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMVADMIN1</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMVADMIN2</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMVADMIN3</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMVADMIN4</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMVADMIN5</key>
  <value>DATE</value>
</entry>
<entry>
  <key>EMVCELLPHONE</key>
  <value>LONG</value>
</entry>
<entry>
  <key>EMVDOUBLON</key>
  <value>LONG</value>
</entry>
<entry>
  <key>EMVHBQ</key>
  <value>DATE</value>
</entry>
<entry>
  <key>EMVISP</key>
  <value>STRING</value>
</entry>
<entry>
  <key>EMVUNROUTABLE</key>
  <value>LONG</value>
</entry>
```

```
<entry>
  <key>FIRSTNAME</key>
  <value>STRING</value>
</entry>
<entry>
  <key>HBQ_REASON</key>
  <value>LONG</value>
</entry>
<entry>
  <key>LASTNAME</key>
  <value>STRING</value>
</entry>
<entry>
  <key>LHE</key>
  <value>LONG</value>
</entry>
<entry>
  <key>MEMBER_ID</key>
  <value>LONG</value>
</entry>
<entry>
  <key>SEED</key>
  <value>LONG</value>
</entry>
<entry>
  <key>SEGMENT</key>
  <value>STRING</value>
</entry>
<entry>
  <key>SMS_DATE_UNJOIN</key>
  <value>DATE</value>
</entry>
<entry>
  <key>SOURCE</key>
  <value>STRING</value>
</entry>
<entry>
  <key>TITLE</key>
  <value>STRING</value>
</entry>
</fields>
</memberTable>
</response>
```

Unsubscribe a Member by Email Address

This method unsubscribes the member(s) matching the given email address.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/unjoinByEmail?token={token}&email={emailAddress}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/unjoinByEmail/{token}/{emailAddress}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID for the unsubscribe
email	The email address		

Error messages

You must fill in the token parameter

You must fill in the email parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/unjoinByEmail?token={token}&email=l=johnsmith@example.com`

REST PI

`https://{server}/apimember/services/rest/member/unjoinByEmail/{token}/johnsmith@example.com`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370855</result>
</response>
```

Unsubscribe a Member by Cellphone Number

This method unsubscribes one or more members who match a given cellphone number.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/unjoinByCellphone?token={token}&cellphone={cellphone}`

REST PI syntax:

`http://{server}/apimember/services/rest/member/unjoinByCellphone/{token}/{cellphone}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID for the unsubscribe
cellphone	The cellphone number of the member		

Error messages
You must fill in the token parameter
You must fill in the email parameter
An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/unjoinByCellphone/token={token}&cellphone=061234567`

REST PI

`https://{server}/apimember/services/rest/member/unjoinByCellphone/{token}/061234567`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Unsubscribe a Member by ID

This method unsubscribes the member matching the given ID.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/unjoinByMemberId?token={token}&id={memberId}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/unjoinByMemberId/{token}/{memberId}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID for the unsubscribe
memberId	The member ID		

Error messages

You must fill in the token parameter

You must fill in the memberId parameter.

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/unjoinByMemberId?token={token}&id=385478`

REST PI

`https://{server}/apimember/services/rest/member/unjoinByMemberId/{token}/385478`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370855</result>
</response>
```

Unsubscribe Member by Value

This method unsubscribes one or more members who match the given criteria.

Note: It should be noted that if the criteria used is the same for multiple members, all of these members will be unsubscribed.

This is a POST method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

`https://{server}/apimember/services/rest/member/unjoinMember/{token}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the insertion or update
memberUID	The field or fields used to identify the member and their values in the following format: <ul style="list-style-type: none">Field1:Value1 Field2:Value2		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Examples

Example 1

Input

URL

`https://{server}/apimember/services/rest/member/unjoinMember/{token}`

HTTP Method

POST

Header

`content-type application/xml`

`accept application/xml`

Body

```
<?xml version="1.0" encoding="UTF-8"?>
  <synchroMember>
```

```
<memberUID>EMAIL:johnsmith@smartfocus.com</memberUID>  
</synchroMember>
```

Output

```
<response responseStatus="success">  
  <result xsi:type="xs:long">370851</result>  
</response>
```

Example 2

Input

URL

https://{server}/apimember/services/rest/member/unjoinMember/{token}

HTTP Method

POST

Header

content-type application/xml

accept application/xml

Body

```
<?xml version="1.0" encoding="UTF-8"?>  
  <synchroMember>  
    <memberUID>FIRSTNAME:John|LASTNAME:Smith</memberUID>  
  </synchroMember>
```

Output

```
<response responseStatus="success">  
  <result xsi:type="xs:long">370851</result>  
</response>
```


Re-Subscribe a Member by Email Address

This method re-subscribes an unsubscribed member using his/her email address. If there are multiple members with the same email address, they will all be re-subscribed.

Note: The number of rejoins per day is limited to 50 per day to avoid massive rejoins and illegal usage of this method. When the limit is reached, you will receive the following error message:
You have reached the maximum number of rejoins per day.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/rejoinByEmail?token={token}&email={emailAddress}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/rejoinByEmail/{token}/{emailAddress}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the resubscription
email	The email address		

Error messages

You must fill in the token parameter
An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/rejoinByEmail/token={token}&email=johnsmith@smartfocus.com`

REST PI

`https://{server}/apimember/services/rest/member/rejoinByEmail/{token}/johnsmith@smartfocus.com`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Re-Subscribe a Member by Cellphone Number

Re-subscribes an unsubscribed member using his/her cell phone number. If there are multiple members with the same number, they will all be re-subscribed.

Note: The number of rejoins per day is limited to 50 per day to avoid massive rejoins and illegal usage of this method. When the limit is reached, you will receive the following error message:
You have reached the maximum number of rejoins per day.

This is a GET method.

REST WADL

`https://{server}/apimember/services/rest?_wadl&_type=xml`

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/rejoinByCellphone?token={token}&email={emailAddress}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/rejoinByCellphone/{token}/{emailAddress}`

Note: Ask your Account Manager for your server name.

Input parameters		Output parameters	
Required parameters	Description		Description
token	The connection token	return	The job ID of the resubscription
cellphone	The cellphone number of the member		

Error messages

You must fill in the token parameter

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/rejoinByCellphone/token={token}&cell-phone=061234567`

REST PI

`https://{server}/apimember/services/rest/member/rejoinByCellphone/{token}/061234567`

Output

```
<response responseStatus="success">  
  <result xsi:type="xs:long">370851</result>  
</response>
```

Re-Subscribe a Member by ID

This method re-subscribes the member matching the given ID.

Note: The number of rejoins per day is limited to 50 per day to avoid massive rejoins and illegal usage of this method. When the limit is reached, you will receive the following error message:
You have reached the maximum number of rejoins per day.

This is a GET method.

REST URL

REST QS syntax:

`https://{server}/apimember/services/rest/member/rejoinByMemberId?token={token}&id={memberId}`

REST PI syntax:

`https://{server}/apimember/services/rest/member/rejoinByMemberId/{token}/{memberId}`

Note: Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
Required parameters			
token	The connection token	return	The job ID of the resubscription
memberId	The ID of the member		

Error messages

You must fill in the token parameter

You have reached the maximum number of rejoins per day.

An error occurred on the server

REST Example

Input

REST QS

`https://{server}/apimember/services/rest/member/rejoinByMemberId?token={token}&id=385478`

REST PI

`https://{server}/apimember/services/rest/member/rejoinByMemberId/{token}/385478`

Output

```
<response responseStatus="success">
  <result xsi:type="xs:long">370851</result>
</response>
```

Reference

WADL

The Web Application Description Language (WADL) is a machine-readable XML-based language that provides a model for describing HTTP-based web applications (such as REST web services).

Web Services

The W3C defines a Web service as a software system designed to support interoperable Machine to Machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. The W3C Web service definition encompasses many different systems, but in common usage the term refers to clients and servers that communicate XML messages that follow the SOAP-standard. Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server, a description in the WSDL. The latter is not a requirement of SOAP endpoint, but it is a prerequisite for automated client-side code generation in the mainstream Java and .NET SOAP frameworks. Some industry organizations, such as the WS-I, mandate both SOAP and WSDL in their definition of a Web service.

WSDL

The Web Services Description Language (WSDL, pronounced 'wiz-dull' or spelled out, 'W-S-D-L') is an XML-based language that provides a model for describing Web services. Version 2.1 has not been endorsed by the World Wide Web Consortium (W3C). Version 2.0, for which several drafts have been released, is expected to become a W3C recommendation. WSDL is an XML-based service description on how to communicate using web services. The WSDL defines services as collections of network endpoints, or ports. WSDL specification provides an XML format for documents for this purpose. WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

XML

The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language. The XML recommendation specifies both the structure of XML, and the requirements for XML processors. XML is considered "general-purpose" because it enables anyone to originate and use a markup language for many types of applications and problem domains. Numerous formally defined markup languages are based on XML, such as RSS, MathML, GraphML, XHTML, Scalable Vector Graphics, MusicXML, and thousands of others. XML's primary purpose is to facilitate the sharing of data across different information systems, particularly systems connected via the Internet. It is a simplified subset of Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible.