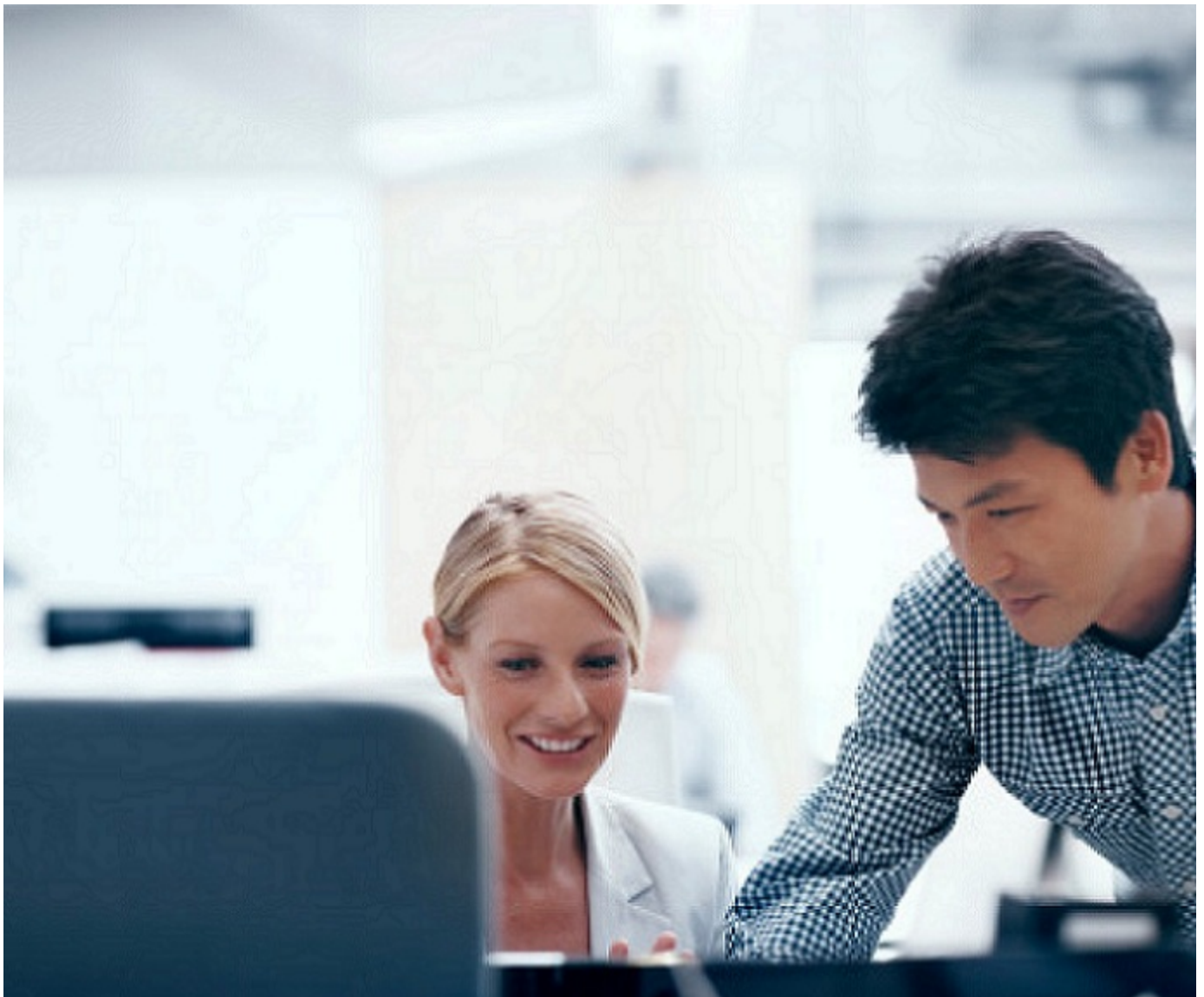


# SmartFocus Cloud Service APIs

<b>Document name</b>	Data Mass Update SOAP API Guide
<b>Service</b>	Data management for mass updates and insertions
<b>Protocol</b>	SOAP over HTTP
<b>Version</b>	10.12
<b>Last updated on</b>	October 20, 2013



## Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>Introduction</b>	<b>4</b>
About This Document	4
About SmartFocus APIs	4
Feedback	4
Support Options	4
Training Options	4
Useful Links	4
Disclaimer	5
<b>Introducing SmartFocus APIs</b>	<b>6</b>
Module Overview	6
<b>Overview of the Data Mass Update API</b>	<b>7</b>
Data Mass Update API Use Cases	7
<b>Getting Started with Integration</b>	<b>8</b>
Prerequisites	8
Quick Start	8
Integration Using APIs	8
Security	9
<b>Connection</b>	<b>10</b>
openApiConnection	11
closeApiConnection	13
<b>uploadFileInsert</b>	<b>14</b>
Description	14
SOAP Example	16
<b>uploadFileMerge</b>	<b>18</b>
Description	18
SOAP Example	20
<b>getLastUpload</b>	<b>22</b>
Description	22
SOAP Example	22
<b>getUploadStatus</b>	<b>24</b>
Description	24
SOAP Example	25
<b>getUploadSummaryList</b>	<b>26</b>
Description	26
SOAP Example	28
<b>getLogFile</b>	<b>30</b>
SOAP Example	30

---

<b>getBadFile</b> .....	<b>32</b>
SOAP Example .....	32
<b>Upload Errors</b> .....	<b>33</b>
<b>Upload Examples</b> .....	<b>34</b>
PHP Examples .....	34
PERL Example .....	36
<b>Reference</b> .....	<b>38</b>

## 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 Data Mass Update SOAP 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](mailto: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.

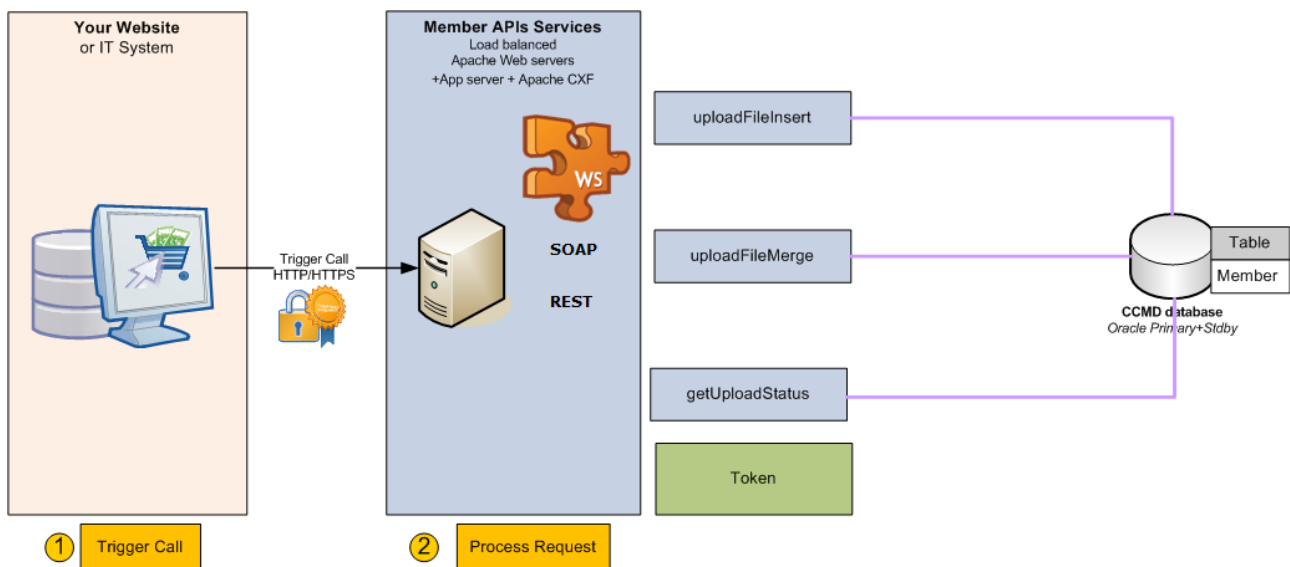
## Introducing SmartFocus APIs

### Module Overview

The Data Mass Update module allows you to create and update your customers' profiles in your SmartFocus member table.

This API offers you a seamless way to insert or merge a file of members through one single API call whenever you want and with the member fields you desire. You can upload up to 5 files at a time, each with a maximum size of 256 MB.

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 Mass Update API

The Data Mass Update API allows you to upload and update mailing lists in your SmartFocus member list.

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

The following methods are available:

Method	Description
<a href="#">openApiConnection</a>	This method provides a session token when given valid credentials.
<a href="#">closeApiConnection</a>	This method terminates the session token.
<a href="#">uploadFileInsert</a>	This method uploads a file containing members and inserts them into the member table.  Note: The maximum file size is 256 Mb. You can upload up to five files simultaneously per SmartFocus account.
<a href="#">uploadFileMerge</a>	This method uploads a file containing members and merges them with those in the member table.  Note: The maximum file size is 256 Mb. You can upload up to five files simultaneously per SmartFocus account.
<a href="#">getLastUpload</a>	This method retrieves the last 20 uploads for the SmartFocus account and their statuses.
<a href="#">getUploadStatus</a>	This method retrieves the status of a file upload.
<a href="#">getUploadSummaryList</a>	This method retrieves a list of uploads and their details.
<a href="#">getLogFile</a>	This method retrieves the log file associated with an upload.
<a href="#">getBadFile</a>	This method retrieves the lines of an uploaded file that could not be uploaded due to errors.

## Data Mass Update API Use Cases

### Upload a Mailing List into the Member List

To upload the member data of a mailing list into the member list:

1. Use the `openApiConnection` method to open the connection.
2. Use the `uploadFileInsert` method to upload the mailing list file.
3. Use the `getUploadStatus` method to obtain the status of the upload.
4. Use the `getLogFile` method to see the details of what was and was not inserted into the member list.
5. Use the `closeApiConnection` method to close the connection.

### Update the Member List

To update the member data in the member list:

1. Use the `openApiConnection` method to open the connection.
2. Use the `uploadFileMerge` method to update the member list with the data in the mailing list file.
3. Use the `getUploadStatus` method to obtain the status of the upload.
4. Use the `getLogFile` method to see the details of what was and was not updated in the member list.
5. Use the `closeApiConnection` 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 (see the *Data Mass Update REST API Guide*)
- SOAP API

### SOAP API

The SOAP API consists in posting an XML file through HTTP POST.

- API call summary:

#### SOAP

##### Submission URL

`https://{server}/apibatchmember/services/BatchMemberService?wsdl`



## SOAP

### 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.
- A parameter and its contents must appear on the same line without spaces and line breaks. In this guide, line breaks have sometimes been added for display reasons.

## 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.

## openApiConnection

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.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

Input parameters	Description	Output parameters	Description
<b>Required parameters</b>			
<b>login</b>	The login provided for API access	<b>return</b>	The token to use in all other API calls
<b>pwd</b>	The password Note: API passwords expire after 365 days.		
<b>key</b>	The manager key copied from SmartFocus (see <a href="#">Connection</a> on page 10)		

### 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.
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!

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:openApiConnection>
      <login>br_test</login>
      <pwd>aptrokez</pwd>
      <key>CdX7Cr1E_26blFNJOsgfdawh6LJ3y6pwg5PEOvA</key>
    </api:openApiConnection>
  </soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:openApiConnectionResponse xmlns:ns2="http://api.service.apibatchmember.emailvision.com/">
      <return>{token}</return>
    </ns2:openApiConnectionResponse>
  </soap:Body>
</soap:Envelope>
```

## closeApiConnection

This method terminates the session token.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

Input parameter	Description	Output parameter	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	<b>return</b>	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

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:closeApiConnection>
      <token>{token}</token>
    </api:closeApiConnection>
  </soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:closeApiConnectionResponse xmlns:n-s2="http://api.service.apibatchmember.emailvision.com/">
      <return>connection closed</return>
    </ns2:closeApiConnectionResponse>
  </soap:Body>
</soap:Envelope>
```

## uploadFileInsert

This method uploads a file containing members and inserts them into the member table.

**Note:** The maximum file size is 256 Mb. You can upload up to five files simultaneously per SmartFocus account.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

## Description

Members from the uploaded file will be inserted into the member table.

The **autoMapping** parameter uses the first line of the file as a mapping. If set to false or omitted the mapping has to be defined in the parameters of the mapping envelope (e.g. column 1 <=> EMAIL, column 2 <=> FIRSTNAME).

The data patterns used for the **dateFormat** parameter are:

- yyyy = Year
- MM = Month
- dd = Day
- HH = Hours (1 -12)
- HH24 = Hours (00-23)
- mi = Minutes (and not mm)
- ss = seconds
- XXX = time zone

**Note:** The data patterns of the date format are not case sensitive. They can be divided by spaces, backslashes (/), colons (:), and hyphens (-).

For example:

- dd/MM/yyyy
- dd/MM/yyyy HH:mi
- MM/dd/yyyy HH24:mi
- yyyy/MM/dd HH24:mi:ss
- dd/MM/yyyy HH:miXXX
- MM/dd/yyyy HH:miXXX
- yyyy/MM/dd HH:miXXX
- dd-MM-yyyy HH:miXXX
- MM-dd-yyyy HH:miXXX
- yyyy-MM-dd HH:miXXX
- dd/MM/yyyy HH:mi:ssXXX
- MM/dd/yyyy HH:mi:ssXXX
- yyyy/MM/dd HH:mi:ssXXX
- dd-MM-yyyy HH:mi:ssXXX
- MM-dd-yyyy HH:mi:ssXXX
- yyyy-MM-dd HH:mi:ssXXX

You can also use the option to deduplicate members in the file before inserting them:

- **dedup**: deduplication envelope parameter
- **criteria**: member field(s) to use as match criteria (e.g. LOWER(EMAIL))
- **order**: **first** or **last** -> the first or last occurrence of a duplicate entry in the file will be used

If you want to skip unsubscribed or quarantine members, **skipUnsubAndHBQ** must be set to **true**.

**Note:**

Files must be passed as SOAP attachments or directly encoded in Base64 in the soap envelope:

```
<file>cid:32922099514</file>
```

or

```
<file>ZW1haWw7Zmly-  
c3RuYW1lO2x-  
hc3RuYW1lO3NvdXJjZTtWYXJjaGFyNTtWYXJjaGFyMTA7VmFyY2hhcjIwO1ZhcmNoYXII...==</file>
```

MTOM (Message Transmission Optimization Mechanism) can be enabled to optimize transfers of binary data. Enabling MTOM varies depending on the SOAP client used (SOAP UI, PHP, Java, PERL, etc.).

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	return	<b>uploadId</b> - The ID of the upload job
<b>file</b>	The content ID of the attachment to upload or the Base64-encoded file content.		
<b>insertUpload</b>	The upload configuration envelope containing the parameters defining the upload.		
<b>fileName</b>	The name of the file to upload		
<b>fileEncoding</b>	The encoding of the file (the default value is UTF-8)		
<b>separator</b>	The separator used: <ul style="list-style-type: none"> <li>• comma (,)</li> <li>• semi-colon (;)</li> <li>• pipe ( )</li> <li>• tab</li> </ul>		
<b>skipFirstLine</b>	Skips the first line in the file (default value is <b>false</b> )		
<b>dateFormat</b>	The date format used in the columns containing dates		
<b>autoMapping</b>	Set to <b>true</b> to automatically map the column headings in the file to the column headings in the database (default value is <b>false</b> ).		
<b>Deduplication Parameters</b>			
<b>dedup</b>	The deduplication envelope parameter which must include <b>criteria</b> , <b>order</b> , and <b>skipUnsubAndHBQ</b> (optional). If not given, deduplication will be set to false.		
<b>criteria</b>	The field to use as the duplication key, e.g. LOWER(EMAIL)		

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>order</b>	Set to <b>first</b> to keep the first occurrence of a duplicate entry in the file. Set to <b>last</b> to keep the last occurrence. Default value is <b>first</b> .		
<b>skipUnsubAndHBQ</b>	Set to <b>true</b> to skip unsubscribed and quarantined members. Omit or set to <b>false</b> to include them.		
<b>Mapping Parameters</b>			
<b>mapping</b>	The mapping envelope parameter containing the column mapping definitions. Required if <b>autoMapping</b> is <b>false</b> .		
<b>column</b>	The column envelope parameter for the mapping parameters: <b>colNum</b> , <b>fieldName</b> , <b>dateFormat</b> (optional), and <b>defaultValue</b> (optional).		
<b>colNum</b>	The number of the column in the file. Required if <b>autoMapping</b> is <b>false</b> .		
<b>fieldName</b>	The column name in the database that should be linked to the specified file column number. Required if <b>autoMapping</b> is <b>false</b> .		
<b>dateFormat</b>	The date format that will override the date format set for the file for the specified column.		
<b>defaultValue</b>	If defined, the values contained in the column for the <b>colNum</b> of the file will be ignored and the defined default value will be inserted.		

Error messages
You must fill in the token parameter
Element 'mapping' is required.
Element 'fileName' is required.
Element 'separator' is required.
Too many uploads are still pending. The limit is 5 uploads by client. You must wait until some uploads are processed.
Element 'separator' is invalid. Allowed are: [, ;   tab].
Upload file is too big. The limit is 256 Mbytes.
Upload file is empty.
An error occurred on the server

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:uploadFileInsert>
      <token>{token}</token>
      <insertUpload>
        <fileName>member.csv</fileName>
        <fileEncoding>UTF-8</fileEncoding>
        <separator>,</separator>
        <skipFirstLine>true</skipFirstLine>
      </insertUpload>
    </api:uploadFileInsert>
  </soapenv:Body>
</soapenv:Envelope>
```



```
<dateFormat>dd/mm/yyyy</dateFormat>
<autoMapping>false</autoMapping>
<dedup>
  <criteria>LOWER(EMAIL)</criteria>
  <order>first</order>
  <skipUnsubAndHBQ>true</skipUnsubAndHBQ>
</dedup>
<mapping>
  <column>
    <colNum>1</colNum>
    <fieldName>EMAIL</fieldName>
  </column>
  <column>
    <colNum>2</colNum>
    <fieldName>FIRSTNAME</fieldName>
  </column>
  <column>
    <colNum>3</colNum>
    <fieldName>DATEOFBIRTH</fieldName>
    <dateFormat>MM/dd/yyyy</dateFormat>
  </column>
  <column>
    <fieldName>SEGMENT</fieldName>
    <defaultValue>TESTAPI</defaultValue>
  </column>
</mapping>
</insertUpload>
<file>cid:834742861923</file>
</api:uploadFileInsert>
</soapenv:Body>
</soapenv:Envelope>
```

## Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:uploadFileInsertResponse xmlns:n-
s2="http://api.service.apibatchmember.emailvision.com/" xmlns:n-
s3="http://exceptions.service.apibatchmember.emailvision.com/">
      <return>{uploadId}</return>
    </ns2:uploadFileInsertResponse>
  </soap:Body>
</soap:Envelope>
```

## uploadFileMerge

This method uploads a file containing members and merges them with those in the member table.

**Note:** The maximum file size is 256 Mb. You can upload up to five files simultaneously per SmartFocus account.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

## Description

Members from the uploaded file will be merged with those in the member table.

- The merge criteria must be specified as a comma-separated list of member fields (these must be defined in the mapping). You may even specify SQL functions.

Examples:

```
<criteria>FIRSTNAME, LASTNAME</criteria>  
<criteria>LOWER (EMAIL) </criteria>
```

- Only the columns that are to be replaced should be defined in the mapping envelopes with the **toReplace** parameter. Other columns will be ignored.
- The data patterns used for the **dateFormat** parameter are:
  - yyyy = Year
  - MM = Month
  - dd = Day
  - HH = Hours (1 -12)
  - HH24 = Hours (00-23)
  - mi = Minutes (and not mm)
  - ss = seconds
  - XXX = time zone

**Note:** The data patterns of the date format are not case sensitive. They can be divided by spaces, backslashes (/), colons (:), and hyphens (-).

For example:

- dd/MM/yyyy
- dd/MM/yyyy HH:mi
- MM/dd/yyyy HH24:mi
- yyyy/MM/dd HH24:mi:ss
- dd/MM/yyyy HH:miXXX
- MM/dd/yyyy HH:miXXX
- yyyy/MM/dd HH:miXXX
- dd-MM-yyyy HH:miXXX
- MM-dd-yyyy HH:miXXX
- yyyy-MM-dd HH:miXXX

- dd/MM/yyyy HH:mi:ssXXX
- MM/dd/yyyy HH:mi:ssXXX
- yyyy/MM/dd HH:mi:ssXXX
- dd-MM-yyyy HH:mi:ssXXX
- MM-dd-yyyy HH:mi:ssXXX
- yyyy-MM-dd HH:mi:ssXXX

**Note:**

Files must be passed as SOAP attachments or directly encoded in Base64 in the soap envelope:

```
<file>cid:32922099514</file>
```

or

```
<file>ZW1haWw7Zmly-  
c3RuYW1lO2x-  
hc3RuYW1lO3NvdXJjZTtWYXJjaGFyNTtWYXJjaGFyMTA7VmFyY2hhcjIwO1ZhcmlNoYXI...==</file>
```

MTOM (Message Transmission Optimization Mechanism) can be enabled to optimize transfers of binary data. Enabling MTOM varies depending on the SOAP client used (SOAP UI, PHP, Java, PERL, etc.).

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	return	<b>uploadId</b> - The ID of the upload job
<b>file</b>	The content ID of the attachment to upload or the Base64-encoded file content.		
<b>mergeUpload</b>	The upload configuration envelope containing the parameters defining the upload.		
<b>fileName</b>	The name of the file to upload		
<b>fileEncoding</b>	The encoding of the file (the default value is UTF-8)		
<b>separator</b>	The separator used: <ul style="list-style-type: none"> <li>• comma (,)</li> <li>• semi-colon (;)</li> <li>• pipe ( )</li> <li>• tab</li> </ul>		
<b>skipFirstLine</b>	Skips the first line in the file (default value is <b>false</b> )		
<b>dateFormat</b>	The date format used in the columns containing dates		
<b>criteria</b>	The field to use as merge criteria, e.g. LOWER(EMAIL)		
<b>Mapping Parameters</b>			
<b>mapping</b>	The mapping envelope parameter containing the column mapping definitions.		
<b>colNum</b>	The number of the column in the file		
<b>fieldName</b>	The column name in the database that should be linked to the specified file column number.		

Input parameter Required parameters	Description	Output parameters	Description
<b>toReplace</b>	Defines whether a field value should or should not be replaced (default value is <b>false</b> ). It must be present for the field value to be replaced.		
<b>dateFormat</b>	The date format that will override the date format set for the file for the specified column.		
<b>defaultValue</b>	If defined, the values contained in the column for the <b>colNum</b> of the file will be ignored and the defined default value will be inserted.		

Error messages
You must fill in the token parameter
Element 'criteria' is required for merge
Element 'mapping' is required.
Element 'fileName' is required.
Element 'separator' is required.
Too many uploads are still pending. The limit is 5 uploads by client. You must wait until some uploads are processed.
Element 'separator' is invalid. Allowed are: [, ;   tab].
Upload file is too big. The limit is 256 Mbytes.
Upload file is empty.
An error occurred on the server

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:uploadFileMerge>
      <token>{token}</token>
      <mergeUpload>
        <fileName>member.csv</fileName>
        <fileEncoding>UTF-8</fileEncoding>
        <separator>,</separator>
        <dateFormat>mm/dd/yyyy</dateFormat>
        <criteria>LOWER(EMAIL)</criteria>
        <mapping>
          <column>
            <colNum>1</colNum>
            <fieldName>EMAIL</fieldName>
          </column>
          <column>
            <colNum>2</colNum>
            <fieldName>FIRSTNAME</fieldName>
            <toReplace>true</toReplace>
          </column>
          <column>
            <colNum>3</colNum>
```

```
        <fieldName>DATEOFBIRTH</fieldName>
        <dateFormat>MM/dd/yyyy</dateFormat>
        <toReplace>true</toReplace>
    </column>
    <column>
        <fieldName>SEGMENT</fieldName>
        <defaultValue>TESTAPI</defaultValue>
    </column>
</mapping>
</mergeUpload>
<file>cid:983851679351</file>
</api:uploadFileMerge>
</soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:uploadFileMergeResponse xmlns:ns2="http://api.service.apibatchmember.emailvision.com/"
xmlns:ns3="http://exceptions.service.apibatchmember.emailvision.com/">
      <return>{uploadId}</return>
    </ns2:uploadFileMergeResponse>
  </soap:Body>
</soap:Envelope>
```

## getLastUpload

This method retrieves the last 20 uploads for the SmartFocus account and their statuses.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

## Description

For each of the 20 retrieved uploads, the following information is provided:

- **id:** The ID of the upload.
- **source:** The source application where the upload was created (API\_BATCH\_MEMBER or CCMD)
- **status:** The status of the upload:
  - **Pending status**
    - STORAGE: Upload file has been successfully uploaded and saved
    - VALIDATED: Upload file has been successfully validated
    - QUEUED: Upload has been queued for processing
    - IMPORTING: Database import is starting
  - **Error status**
    - ERROR: File validation has failed
    - FAILURE: Database import has failed
  - **Final status**
    - DONE: Upload is complete
    - DONE WITH ERRORS: Upload is complete but there were some errors

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	return	<b>uploads</b> - a list of the last 20 uploads

### Error messages

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

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
```

```
<soapenv:Header/>
<soapenv:Body>
  <api:getLastUpload>
    <token>{token}</token>
  </api:getLastUpload>
</soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getLastUploadResponse xmlns:ns2="http://api.service.apibatchmember.emailvision.com/"
      xmlns:ns3="http://exceptions.service.apibatchmember.emailvision.com/">
      <return>
        <id>197258</id>
        <source>API_BATCH_MEMBER</source>
        <status>QUEUED</status>
      </return>
      <return>
        <id>197257</id>
        <source>API_BATCH_MEMBER</source>
        <status>DONE</status>
      </return>
      <return>
        <id>197037</id>
        <source>CCMD</source>
        <status>DONE</status>
      </return>
    </ns2:getLastUploadResponse>
  </soap:Body>
</soap:Envelope>
```

## getUploadStatus

This method retrieves the status of a file upload.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

## Description

The process of an upload job is divided into several steps. Each step is associated with a specific status.

This method provides the current status of an upload job.

### Pending status

- **STORAGE:** Upload file has been successfully uploaded and saved
- **VALIDATED:** Upload file has been successfully validated
- **QUEUED:** Upload has been queued for processing
- **IMPORTING:** Database import is starting

### Error status

- **ERROR:** File validation has failed
- **FAILURE:** Database import has failed

### Final status

- **DONE:** Upload is complete
- **DONE WITH ERRORS:** Upload is complete but there were some errors

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	return	<b>status</b> - the status of the upload job
<b>uploadId</b>	The ID of the upload job		

Error messages
You must fill in the token parameter
You must fill in the uploadId parameter.
No flatUpload found !
An error occurred on the server



## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:getUploadStatus>
      <token>{token}</token>
      <uploadId>{uploadId}</uploadId>
    </api:getUploadStatus>
  </soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getUploadStatusResponse xmlns:ns2="http://api.service.apibatchmember.emailvision.com/">
      <return>
        <status>QUEUED</status>
      </return>
    </ns2:getUploadStatusResponse>
  </soap:Body>
</soap:Envelope>
```

## getUploadSummaryList

This method retrieves a list of uploads and their details.

### WSDL Location

`http://{server}/apibatchmember/services/BatchMemberService?wsdl`

**Note:** Ask your Account Manager for your server name.

## Description

For each of the retrieved uploads, the following information is provided:

- **id:** The ID of the upload.
- **source:** The source application where the upload was created (API\_BATCH\_MEMBER or CCMD)
- **status:** The status of the upload:
  - **Pending status**
    - STORAGE: Upload file has been successfully uploaded and saved
    - VALIDATED: Upload file has been successfully validated
    - QUEUED: Upload has been queued for processing
    - IMPORTING: Database import is starting
  - **Error status**
    - ERROR: File validation has failed
    - FAILURE: Database import has failed
  - **Final status**
    - DONE: Upload is complete
    - DONE WITH ERRORS: Upload is complete but there were some errors
- **manager:** The manager (login) who launched the upload
- **type:** The type of upload:
  - **merge:** The members in the uploaded file were merged with the members in the member table (i.e. if the member table already contains a member with an email address from the uploaded file, this member's details will be updated).
  - **insert:** The members in the uploaded file were entered as new members in the member table.
- **date:** The date of the upload
- **name:** The name of the uploaded file
- **size:** The size of the upload

Input parameter Required parameters	Description	Output parameters	Description
token	The connection token	return	The uploads and their details
<b>List Options Parameters</b>			
page	The page to return		
pageSize	The number of elements to return per page (default: 1000)		
<b>Search Parameters</b>			
search	The search envelope		
uploadId	The ID of the upload job		
minCreatedDate	<p>The start date of the creation date range.</p> <p>The date format follows the ISO 8601 rules where date and time values are ordered from the most to the least significant.</p> <p>Example:</p> <ul style="list-style-type: none"> <li>2013-04-05</li> <li>2013-04-05+02:00</li> <li>2013-04-05T10:20:58</li> <li>2013-04-05T10:20:58+02:00</li> </ul> <p>Note: It is highly recommended to always include the time zone. If the time zone is omitted, the SmartFocus server time zone will be used.</p>		
maxCreatedDate	<p>The end date of the creation date range.</p> <p>The date format follows the ISO 8601 rules where date and time values are ordered from the most to the least significant.</p> <p>Example:</p> <ul style="list-style-type: none"> <li>2013-04-05</li> <li>2013-04-05+02:00</li> <li>2013-04-05T10:20:58</li> <li>2013-04-05T10:20:58+02:00</li> </ul> <p>Note: It is highly recommended to always include the time zone. If the time zone is omitted, the SmartFocus server time zone will be used.</p>		
source	The source application where the upload was created (API_BATCH_MEMBER or CCMD)		

Input parameter Required parameters	Description	Output parameters	Description
status	<p>The status of the upload:</p> <ul style="list-style-type: none"> <li> <b>Pending status</b> <ul style="list-style-type: none"> <li>STORAGE: Upload file has been successfully uploaded and saved</li> <li>VALIDATED: Upload file has been successfully validated</li> <li>QUEUED: Upload has been queued for processing</li> <li>IMPORTING: Database import is starting</li> </ul> </li> <li> <b>Error status</b> <ul style="list-style-type: none"> <li>ERROR: File validation has failed</li> <li>FAILURE: Database import has failed</li> </ul> </li> <li> <b>Final status</b> <ul style="list-style-type: none"> <li>DONE: Upload is complete</li> <li>DONE WITH ERRORS: Upload is complete but there were some errors</li> </ul> </li> </ul>		
<b>Sort Options Parameters</b>			
sortOptions	The envelope containing the sortOption envelope(s)		
sortOption	The sortOption envelope that specifies which column should be used for the sort and in which order the sort should be applied		
column	The column that should be used for the sort		
order	<p>The order of the sort (i.e. ascending or descending):</p> <ul style="list-style-type: none"> <li>ASC</li> <li>DES</li> </ul>		

#### Error messages

You must fill in the token parameter

An error occurred on the server

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:getUploadSummaryList>
      <token>{token}</token>
      <listOptions>
```

```
<page>1</page>
<pageSize>10</pageSize>
<search>
  <source>ccmd</source>
  <status>complete</status>
</search>
<sortOptions>
  <sortOption>
    <column>date</column>
    <order>ASC</order>
  </sortOption>
</sortOptions>
</listOptions>
</api:getUploadSummaryList>
</soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getUploadSummaryListResponse xmlns:n-
s2="http://api.service.apibatchmember.emailvision.com/" xmlns:n-
s3="http://exceptions.service.apibatchmember.emailvision.com/">
      <return>
        <uploadSummaries>
          <uploadSummaryEntity>
            <uploadId>170</uploadId>
            <manager>John Smith</manager>
            <source>ccmd</source>
            <type>merge</type>
            <date>2013-05-10T14:48:07+02:00</date>
            <name>API2013_05_10_14_48_07</name>
            <size>26993</size>
            <status>complete</status>
          </uploadSummaryEntity>
          <uploadSummaryEntity>
            <uploadId>171</uploadId>
            <manager>John Smith</manager>
            <source>ccmd</source>
            <type>insert</type>
            <date>2013-05-10T14:48:22+02:00</date>
            <name>API2013_05_10_14_48_22</name>
            <size>26993</size>
            <status>complete</status>
          </uploadSummaryEntity>
        </uploadSummaries>
        <page>1</page>
        <pageSize>10</pageSize>
        <nbTotalItems>2</nbTotalItems>
        <nextPage>false</nextPage>
        <previousPage>false</previousPage>
      </return>
    </ns2:getUploadSummaryListResponse>
  </soap:Body>
</soap:Envelope>
```

## getLogFile

This method retrieves the log file associated with an upload.

### WSDL Location

<http://{server}/apibatchmember/services/BatchMemberService?wsdl>

**Note:** Ask your Account Manager for your server name.

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	return	The logs for the upload
<b>uploadId</b>	The ID of the upload job		

### Error messages

You must fill in the token parameter

No flatUpload found!

An error occurred on the server

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:getLog>
      <token>{token}</token>
      <uploadId>{uploadId}</uploadId>
    </api:getLog>
  </soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getLogFileResponse xmlns:ns2="http://api.service.apibatchmember.emailvision.com/"
      xmlns:ns3="http://exceptions.service.apibatchmember.emailvision.com/">
      <return>SQL*Loader: Release 10.2.0.1.0 - Production on Thu Jul 28 11:46:10 2011
```

Copyright (c) 1982, 2005, Oracle. All rights reserved.

Control File: /var/emv/DATA\_UPLOAD/490/LUR\_ACCOUNT\_529.ctl  
Character Set UTF8 specified for all input.  
Using character length semantics.

Data File: /var/emv/DATA\_UPLOAD/490/LUR\_ACCOUNT\_529.dat  
Bad File: /var/emv/DATA\_UPLOAD/490/LUR\_ACCOUNT\_529.bad  
Discard File: none specified

(Allow all discards)

Number to load: ALL  
Number to skip: 1  
Errors allowed: 1  
Bind array: 64 rows, maximum of 256000 bytes  
Continuation: none specified  
Path used: Conventional

Table TEMP\_FU\_529, loaded from every logical record.  
Insert option in effect for this table: APPEND  
TRAILING NULLCOLS option in effect

Column Name	Position	Len	Term	Encl	Datatype
CLIENT_ID					CONSTANT
Value is '490'					
EMAIL	FIRST	765	;	O(")	CHARACTER
SQL string for column : "DECODE(REGEXP_INSTR(:EMAIL,'.+@.+.+',1),0,NULL,LOWER(TRIM(:EMAIL)))"					
TEMPORARY_MEMBER_ID	NEXT	*	;	O(")	CHARACTER
SQL string for column : "DECODE(:TEMPORARY_MEMBER_ID,null, SEQ_TMP_529.nextval, SEQ_TMP_529.nextval)"					

Table TEMP\_FU\_529:  
0 Rows successfully loaded.  
0 Rows not loaded due to data errors.  
0 Rows not loaded because all WHEN clauses were failed.  
0 Rows not loaded because all fields were null.

Space allocated for bind array: 65920 bytes(64 rows)  
Read buffer bytes: 1048576

Total logical records skipped: 1  
Total logical records read: 0  
Total logical records rejected: 0  
Total logical records discarded: 0

Run began on Thu Jul 28 11:46:10 2011  
Run ended on Thu Jul 28 11:46:10 2011

Elapsed time was: 00:00:00.13  
CPU time was: 00:00:00.02</return>  
</ns2:getLogFileResponse>  
</soap:Body>  
</soap:Envelope>

## getBadFile

This method retrieves the lines of an uploaded file that could not be uploaded due to errors.

### WSDL Location

<http://{server}/apibatchmember/services/BatchMemberService?wsdl>

**Note:** Ask your Account Manager for your server name.

Input parameter	Description	Output parameters	Description
<b>Required parameters</b>			
<b>token</b>	The connection token	return	The lines that could not be uploaded
<b>uploadId</b>	The ID of the upload job		

### Error messages

You must fill in the token parameter

No flatUpload found!

An error occurred on the server

## SOAP Example

### Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:api="http://api.service.apibatchmember.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:getBadFile>
      <token>{token}</token>
      <uploadId>{uploadId}</uploadId>
    </api:getBadFile>
  </soapenv:Body>
</soapenv:Envelope>
```

### Output

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:getBadFileResponse xmlns:ns2="http://api.service.apibatchmember.emailvision.com/">
      <return>ccommanderqa@hotmail.co.uk      bob16      sinclar16      33600000015
16/02/2010^@0906/07/1984      QA
      ccommanderqa3@yahoo.com.au      bob26      sinclar26      33600000025
26/08/2010^@0916/02/1973      QA
      testm30@wanadoo.fr      bob17      sinclar17      33600000016      17/02/2010
08/04/1 87      QA
      ccommanderqa@fastmail.co.uk      bob27      sinclar27      33600000026
27/08/2010^@0917/06/1972      QA</return>
    </ns2:getBadFileResponse>
  </soapenv:Body>
</soapenv:Envelope>
```



## Upload Errors

When an error occurs, the response will contain an object that contains 3 elements:

- status: INVALID\_FILE, INVALID\_PARAMETERS, MAX\_NB\_UPLOADS, INVALID\_UPLOAD\_ID, GET\_MANAGER\_FAILED, GET\_STATUS\_FAILED, GET\_CLIENT\_FAILED, CREATE\_JOB\_FAILED
- description: short description of the problem
- fields: parameters that are the source of the problem

Parameter Errors	
INVALID_FILE	<ul style="list-style-type: none"> <li>• Upload file is empty</li> <li>• Upload file is too big. The limit is 250 M bytes.</li> </ul>
INVALID_PARAMETERS	<ul style="list-style-type: none"> <li>• Element 'mapping' is required.</li> <li>• Element 'fileName' is required.</li> <li>• Element 'separator' is required.</li> <li>• Element 'separator' is invalid. Allowed are: [, ;   tab]</li> <li>• Element 'criteria' is required for merge</li> </ul>
MAX_NB_UPLOADS	Too many uploads are still pending. You must wait until some uploads are processed.

Server Errors	
GET_MANAGER_FAILED	An error happened on the server while retrieving the manager from the database.
GET_STATUS_FAILED	An error happened on the server while retrieving the status from the database.
GET_CLIENT_FAILED	An error happened on the server while retrieving the client from the database.
CREATE_JOB_FAILED	An error happened on the server while saving the upload job to the database.

Example:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Fault occurred while processing.</faultstring>
      <detail>
        <ns3:BatchMemberServiceException xmlns:n-
s2="http://api.service.apibatchmember.emailvision.com/" xmlns:n-
s3="http://exceptions.service.apibatchmember.emailvision.com/">
          <description>No flatUpload found!</description>
          <fields>123</fields>
          <status>GET_FLATUPLOAD_FAILED</status>
        </ns3:BatchMemberServiceException>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

## Upload Examples

### PHP Examples

#### Insert Upload

```
<?php

$login = "loginAPI";
$password = "passwordAPI";
$key = "CdX7Cr1K-EONkElTcdIm0ZaSah4rRfPB-AjtSqvdKPvpJpU";
$fileName = "file/insertMember.txt";
$batchMemberWsdUrl="https://<server>/apibatchmember/services/BatchMemberService?wsdl"

try {

    // Create SOAP client
    $client = new WSClient(array("wsdl" => $batchMemberWsdUrl, "useMTOM" => TRUE));

    $proxy = $client->getProxy();

    // OPEN CONNECTION

    $loginParameters['login'] = $login;
    $loginParameters['pwd'] = $password;
    $loginParameters['key'] = $key;

    $response = $proxy->openApiConnection($loginParameters);

    $token = $response['return'];

    print "openApiConnection ok: token=".$token."<br>";

    // UPLOAD FILE INSERT

    $uploadParameters['token'] = $token;
    $uploadParameters['insertUpload']['fileName'] = "insertMember.txt";
    $uploadParameters['insertUpload']['fileEncoding'] = "UTF-8";
    $uploadParameters['insertUpload']['separator'] = ",";
    $uploadParameters['insertUpload']['autoMapping'] = true;
    $uploadParameters['file'] = file_get_contents($fileName);

    $response2 = $proxy->uploadFileInsert($uploadParameters);
    $uploadId = $response2['return'];

    print "uploadFileInsert ok: uploadId=".$uploadId."<br> token=".$token."<br>";

    // UPLOAD STATUS

    sleep(10);

    $statusParameters['token'] = $token;
    $statusParameters['uploadId'] = $uploadId;

    $response3 = $proxy->getUploadStatus($statusParameters);

    $status = $response3['return']['status'];

    print "getUploadStatus ok: status=".$status."<br>";

} catch(Exception $e) {
    print "Exception thrown: ".$e."<br>";
}
```

```
}  
?>
```

## Merge Upload

```
<?php
```

```
$login = "loginAPI";  
$password = "passwordAPI";  
$key = "CdX7Cr1K-EONkElTcdIm0ZaSah4rRfPB-AjtSqvdKPvpJpU";  
$fileName = "file/mergeMember.txt";  
$batchMemberWsdUrl="https://<server>/apibatchmember/services/BatchMemberService?wsdl"  
  
try {  
  
    // Create SOAP client  
    $client = new WsClient(array("wsdl" => $batchMemberWsdUrl, "useMTOM" => TRUE));  
  
    $proxy = $client->getProxy();  
  
    // OPEN CONNECTION  
  
    $loginParameters['login'] = $login;  
    $loginParameters['pwd'] = $password;  
    $loginParameters['key'] = $key;  
  
    $response = $proxy->openApiConnection($loginParameters);  
  
    $token = $response['return'];  
  
    print "openApiConnection ok: token=".$token."</br>";  
  
    // UPLOAD FILE MERGE  
  
    $uploadParameters['token'] = $token;  
    $uploadParameters['mergeUpload']['fileName'] = "mergeMember.txt";  
    $uploadParameters['mergeUpload']['fileEncoding'] = "UTF-8";  
    $uploadParameters['mergeUpload']['separator'] = ";";  
    $uploadParameters['mergeUpload']['criteria'] = "LOWER(EMAIL)";  
    $uploadParameters['mergeUpload']['skipFirstLine'] = true;  
    $uploadParameters['file'] = file_get_contents($fileName);  
  
    // Field criteria  
    $column1['colNum'] = "1";  
    $column1['fieldName'] = "EMAIL";  
  
    // Field to update  
    $column2['colNum'] = "2";  
    $column2['fieldName'] = "FIRSTNAME";  
    $column2['toReplace'] = "true";  
  
    // Field with default value  
    $column3['colNum'] = "3";  
    $column3['fieldName'] = "SEGMENT";  
    $column3['defaultValue'] = "TEST_API";  
  
    $uploadParameters['mergeUpload']['mapping']['column'] = array($column1, $column2, $column3);  
  
    $response2 = $proxy->uploadFileMerge($uploadParameters);  
    $uploadId = $response2['return'];  
  
    print "uploadFileMerge ok: uploadId=".$uploadId."</br>";  
  
    // UPLOAD STATUS
```

```
        sleep(10);

        $statusParameters['token'] = $token;
        $statusParameters['uploadId'] = $uploadId;

        $response3 = $proxy->getUploadStatus($statusParameters);

        $status = $response3['return']['status'];
        print "getUploadStatus ok: status=".$status."</br>";

    } catch(Exception $e) {
        print "Exception thrown: ".$e."</br>";
    }

    ?>
```

### PERL Example

```
use SOAP::Lite;
use MIME::Base64 qw(encode_base64);

$login = "loginAPI";
$password = "passwordAPI";
$key="CdX7Cr1K-EONkElTcdIm0ZaSah4rRfPB-AjtSqvdKPvpJpU";

# Service details
$NAMESPACE = 'http://api.service.apibatchmember.emailvision.com/';
$ENDPOINT = 'http://192.168.3.8/apibatchmember/services/BatchMemberService';

# Create interface to the service
$service = new SOAP::Lite(uri => $NAMESPACE, proxy => $ENDPOINT);

$service->autotype(1);
$service->readable(1);
$service->soapversion('1.1');
$service->envprefix('SOAP-ENV');
$service->ns($NAMESPACE);
$service->default_ns($NAMESPACE);

# Add a fault handler to map a fault to a die
$service->on_fault(
    sub {      # SOAP fault handler
        my $soap = shift;
        my $res = shift;

        # Map faults to exceptions
        if ( ref($res) eq '' ) {
            die($res);
        }
        else {
            die( $res->faultstring );
        }
        return new SOAP::SOM;
    }
);

# Open API Connection

$loginParameter = SOAP::Data->new(name => 'login', value => $login)->attr({xmlns => ""});
$passwordParameter = SOAP::Data->new(name => 'pwd', value => $password)->attr({xmlns => ""});
$keyParameter = SOAP::Data->new(name => 'key', value => $key)->attr({xmlns => ""});
```

```
my $result = $service->openApiConnection($loginParameter, $passwordParameter, $keyParameter) ;

unless ($result->fault) {
    $token = $result->result();
    print "token: ";
    print $token;
    print "\n";

} else {
    print join ', ', $result->faultcode, $result->faultstring;
    exit;
}

$tokenParameter = SOAP::Data->new(name => 'token', value => $token, type=>'')->attr({xmlns =>
''});

# Insert upload

$insertUploadParameter = SOAP::Data->name('insertUpload' =>
\SOAP::Data->value(
    SOAP::Data->name('fileName' => 'fichier.csv')->attr({xmlns => ''}),
    SOAP::Data->name('fileEncoding' => 'UTF-8')->attr({xmlns => ''}),
    SOAP::Data->name('separator' => ',')->attr({xmlns => ''}),
    SOAP::Data->name('autoMapping' => true)->attr({xmlns => ''})
)->attr({xmlns => ''})
)->attr({xmlns => ''});

# File attachment

open(FILE, "import.txt") || die("Could not open file!");
$fileContent = <FILE>;
$fileParameter = SOAP::Data->name('file' => encode_base64($fileContent))->attr({xmlns => ''});
$resultUpload = $service->uploadFileInsert($tokenParameter, $insertUploadParameter, $fileParameter);

print "Upload ID: ";
print $resultUpload->result();
print "\n";

print "Done\n";
```

## 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.