WSMailUpImport.StartImportProcesses

This method creates and automatically starts an import process for the contacts listed in the xmlDoc parameter. You can use this method instead of calling a sequence of NewImportProcess and StartProcess methods. *StartImportProcesses* can also be used to update fields of an existing contact.

Method parameters

Mandatory fields must be specified even when they are empty. Optional fields can be skipped if you want to use the default value.

Parameter	
listsIDs *string*	**Mandatory** - List identifiers. You can include multiple list IDs, separated by semicolons. *(See "Lists and Groups" section below for more information and examples.)* Example: `1;2;3`
listsGUIDs *string*	**Mandatory** - List GUIDs. You can include multiple list GUIDs, separated by semicolons. *(See "Lists and Groups" section below for more information and examples.)* Example: `abc-123-def-456;ghi-789-jkl-123;mno-456-pqr-789`
xmlDoc *string*	**When specified:** An XML string containing the recipients to be imported to the specified lists and groups. In this case the method Does not import any pending lists that were previously submitted using NewImportProccess **If not specified (NULL):** method sequentially imports all pending XML structures that have been previously submitted using NewImportProccess. This option is available only starting from MailUp 8.2.1 or higher *See "Best Practices" and "XML Structure" sections below for known restrictions, more information and examples.*
groupsIDs *string*	**Mandatory** - Group identifiers for each list, separated by semicolons. You can specify multiple groups for each list as well, separated by commas. *(See "Lists and Groups" section below for more information and examples.)* Example: `22,23;24;25`
importType *integer*	**Mandatory** - Import type. Any empty field is ignored if importType is equal to 1,2 or 3 • 1 = import occur only on email channel (mobile number, if present, is discarded) • 2 = subscriptions occur only on SMS channel (email address, if present, is discarded) • 3 = subscriptions occur on both email and SMS channels (no field is discarded) Any empty field overwrites the value on MailUp if importType is equal to 4, 5 or 6 • 4 = import occur only on email channel (mobile number, if present, is discarded) • 5 = subscriptions occur only on SMS channel (email address, if present, is discarded) • 6 = subscriptions occur on both email and SMS channels (no field is discarded)
mobileInput Type integer	**Mandatory** - Mobile number input type. • 1 = Entire number in a single field • 2 = Prefix and Number separated into different fields *(See "XML Structure" section below for more information and examples.)*
asPending *boolean*	**Mandatory** - If *"true"* it subscribes (or unsubscribes, if *"asOptOut=true"*) also the recipients that were pending in specified MailUp list before import. Please note that the name of this parameter is misleading, see table in Best Practices for details *(Default = false)*

ConfirmEmail *boolean*	**Mandatory** - If *"true"* sets subscription status of specified recipients as "pending" and sends them a confirmation email. Status change to "pending" and email sending applies to either brand new recipients or to recipients that were already subscribed or pending to the specified list before import. See table in Best Practices for details. *(Default = false)*
asOptOut *boolean*	**Mandatory** - Imports recipients as "unsubscribed" when set to *"true"*, regardless if they were previously subscribed into specified MailUp list. If you want to cancel subscription (i.e. unsubscribe) also for the recipients that were previously pending, you should set both "asOptOut=true" and "asPending=true". See table in Best Practices for details. *(Default = false)*
forceOptIn *boolean*	**Mandatory** - If *"true"* it enables change of subscription status even if a recipient was "unsubscribed" before import. It can be used to force "subscribed" or "pending" status for previously unsubscribed recipients, please note that this status change does not apply when recipient was automatically unsubscribed due to hard bounce or complaints feedback loop. See table in Best Practices for details. *(Default = false)*
replaceGroups *boolean*	**Mandatory** - Replace existing groups when set to *"true"*. The system will automatically remove previously subscribed groups and keep only the groups specified in the `groupsIDs` parameter. *(Default = false)*

The following table shows the most common combinations for the parameters that manage subscription status of imported recipients that are already present on MailUp

Action required on already existing recipients	**ConfirmEmail**	**AsOptout**	**AsPending**	**ForceOptIn**	
No changes on subscription status	False	False	False	False	Default case
Pending become Subscribed	False	False	True	False	It's a bad practice to force subscription of pending recipients at import time, the recommended practice with double optin is that you import as pending and you let MailUp do this status change when recipient actually clicks on the confirmation link.
Unsubscribed become Subscribed	False	False	False	True	Except for recipients that were unsubscribed due to hard bounce or complaints feedback loop
Subscribed become Pending	True	False	False	False	With this configuration the confirmation email is sent to both previously subscribed and previously pending
Unsubscribed become Pending	True	False	Any	True	Except for recipients that were unsubscribed due to hard bounce or complaints feedback loop Despite of its name, "*AsPending*" value is ignored here.
Subscribed become Unsubscribed	False	True	False	False	
Pending become Unsubscribed	False	True	True	False	

Best practices

1. Method call fails when another import process is running. You can use GetProcessDetails method to check if a previously started process is still running; while an alternative solution consist of periodically retry the method call until it returns ReturnCode=0
2. Method response contains either a global ReturnCode or a specific ReturnCode for each import process that has been started by this method. Call is successful only when all ReturnCodes are equal to "0"
3. There is not a known size limit for xmlDoc field: it has been successfully tested with 1.2 million characters (in this case return value is quite immediate but the whole import process could take several minutes and other import requests will be denied during this period)
4. When using cloud services it can happen that your application must respect a size limit in API calls that is lower than the size of the XML that you would pass as xmlDoc parameter. In this case you can split your import data, call several times NewImportProcess with smaller recipient lists and then use StartImportProcesses (with empty xmlDoc parameter) to sequentially start all previously created import processes. This behavior requires MailUp 8.2.1 or higher. Please note that, when combining one or more calls to NewImportProcess and a final call to StartImportProcesses , the parameters of StartImportProcesses do not overwrite the import settings that were specified by each NewImportProcess request
5. With ConfirmEmail=true, MailUp automatically selects the confirmation request message (you can customize it on Settings > List Settings > Notifications > Confirmation request) and it creates a queue of recipients that will receive that message. Please note that queue creation takes a while (even a couple of minutes if you have 1M recipients to be imported and notified) and queued message is made "ready for immediate sending" but it IS NOT automatically sent to the recipients. To send a queued message you can use StartDelivery method. You could also use GetNewsletterQueues before calling StartDelivery in order to double check if queuing in "ImmediateSendingQueue" is completed.

Lists and Groups

You must specify either `listIDs` or `listsGUIDs` parameters. If you specify both fields, then the number of items in each parameter must match, and you must use semicolons to delineate empty values. We've provided examples below to help with understanding how to work with these parameters.

Add recipients to a single list (no groups)

In this example values are provided for both `listsIDs` and `listsGUIDs` parameters (mandatory fields). There are no groups, so we include an empty `groupsIDs` tag.

```
<ws:listsIDs>1</ws:listsIDs>
<ws:listsGUIDs>66af9900-7dd7-4cca-9125-beadaf3a3a59</ws:listsGUIDs>
<ws:groupsIDs></ws:groupsIDs>
```

Add recipients to a single list (single group)

In this example values are provided for both `listsIDs` and `listsGUIDs` parameters (mandatory fields), and since there is a single group we include it in the `groupsIDs` parameter.

```
<ws:listsGUIDs>66af9900-7dd7-4cca-9125-beadaf3a3a59</ws:listsGUIDs>
<ws:groupsIDs>22</ws:groupsIDs>
```

Add recipients to a single list (multiple groups)

In this example values are provided for both `listsIDs` and `listsGUIDs` parameters (mandatory fields). There are multiple groups for this list, so we separate them with a comma in the `groupsIDs` parameter.

```
<ws:listsIDs>1</ws:listsIDs>
<ws:listsGUIDs>66af9900-7dd7-4cca-9125-beadaf3a3a59</ws:listsGUIDs>
<ws:groupsIDs>22,23</ws:groupsIDs>
```

Add recipients to multiple lists (no groups)

In this example values are provided for both `listsIDs` and `listsGUIDs` parameters (mandatory fields). Each of the parameters must have the same number of elements *(separated by semi-colons)*, so since there are no groups for either list, we include a single semi-colon in the `groupsID` parameter to denote that there are two elements.

```
<ws:listsIDs>1;2</ws:listsIDs>
<ws:listsGUIDs>66af9900-7dd7-4cca-9125-beadaf3a3a59;0e591119-xxxx-yyyy-zzzz-6ac75384b564</ws:listsGUIDs>
<ws:groupsIDs>;</ws:groupsIDs>
```

Add recipients to multiple lists (one group per list)

In this example values are provided for both `listsIDs` and `listsGUIDs` parameters (mandatory fields). Each of the parameters must have the same number of elements *(separated by semi-colons)*. In this case, we are also specifying group 22 for list 1, and group 13 for list 2.

```
<ws:listsIDs>1;2</ws:listsIDs>
<ws:listsGUIDs>66af9900-7dd7-4cca-9125-beadaf3a3a59;0e591119-xxxx-yyyy-zzzz-6ac75384b564</ws:listsGUIDs>
<ws:groupsIDs>22;13</ws:groupsIDs>
```

Add recipients to multiple lists (multiple groups)

In this example values are provided for both `listsIDs` and `listsGUIDs` parameters (mandatory fields). The first list *(ID=1)* has two groups *(22, 23)*, so we use a comma to separate them in the first element. We then use a semi-colon to denote the second element, since the groupsIDs, listsIDs and listsGUIDs parameters must all have the same number of elements

```
<ws:listsIDs>1;2</ws:listsIDs>
<ws:listsGUIDs>66af9900-7dd7-4cca-9125-beadaf3a3a59;0e591119-xxxx-yyyy-zzzz-6ac75384b564</ws:listsGUIDs>
<ws:groupsIDs>22,23;13</ws:groupsIDs>
```

XML Structure

The XML structure for each recipient needs to be consistent for all subscribers, and include empty tags for required values that are empty. When specifying the phone number for a recipient, the structure of your XML must match the `mobileInputType` parameter, where either the entire phone number is represented in a single attribute, or the prefix and number are represented in separate attributes.

For example, if the `mobileInputType` parameter is set to `1`,use the following XML structure:

```
<!--Option 1: number and prefix in a single field (use mobileInputType=1)-->
<subscriber email="user@myprovider.com" Prefix="" Number="+0018889624587" Name="">
```

If the `mobileInputType` parameter is set to `2`, use the following XML structure:

```
<!--Option 2: number and prefix in separate fields (use mobileInputType=2)-->
<subscriber email="user@myprovider.com" Prefix="+001" Number="8889624587" Name="">
```

In case you also need to specify personal data fields an example is provided below

- Personal data fields shall be specified in **progressive order** and you shall also include empty fields. It is also recommended to use the same data structure (i.e. **the same number of fields for each record**, even if some fields are empty) for all subscribers. In case of update of an existing subscriber, the empty fields are handled as "don't change this field" when you specify 1, 2 or 3 as **ImportType.**
- Do not exceed 50 characters for "email" field and 100 characters for "campo" fields (up to 200 characters are allowed if you use only 7-bit ASCII strings)
- If you want to update an existing subscriber and clear one or more of its fields you shall use import type with value 4, 5, or 6; in this case any empty field in xmlDoc parameter resets the correspondent field on MailUp console account.

```
<subscribers>
  <subscriber email="mike@example.com" Prefix="" Number="" Name="">
    <campo1>Mike</campo1>
    <campo2>Brown</campo2>
    <campo3>Example Company</campo3>
    <campo4>Los Angeles</campo4>
    <campo5> </campo5>
    <campo6>90125</campo6>
    <campo7>CA</campo7>
    <campo8>US</campo8>
    <campo9>555 Some Street</campo9>
    <campo10></campo10>
    <campo11>555-123-1234</campo11>
  </subscriber>
  <!-- repeat for each recipient to import -->
</subscribers>
```

You can use `0` and `1` in place of `true` and `false` for boolean parameter values.

Unsupported characters in subscribers fields

All field values are handled as strings, character '|' (pipe) is not allowed and may lead to "-402" error codes

SOAP Examples

An example SOAP request to `StartImportProcesses`:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://ws.
mailupnet.it/">
  <soapenv:Header>
    <ws:Authentication>
      <ws:User>user</ws:User>
      <ws:Password>password</ws:Password><ws:encType>UTF-8</ws:encType>
    </ws:Authentication>
  </soapenv:Header>
  <soapenv:Body>
    <ws:StartImportProcesses>
      <ws:listsIDs>1</ws:listsIDs>
      <ws:listsGUIDs>66af9900-xxxx-yyyy-zzzz-beadaf3a3a59</ws:listsGUIDs>
      <ws:xmlDoc>
        &lt;subscribers&gt;&lt;subscriber email=&quot;mike@example.com&quot; Prefix=&quot;&quot;
Number=&quot;&quot; Name=&quot;&quot;&gt;&lt;campo1&gt;Mike&lt;/campo1&gt;&lt;campo2&gt;Brown&lt;
/campo2&gt;&lt;campo3&gt;Example Company&lt;/campo3&gt;&lt;campo4&gt;Los Angeles&lt;/campo4&gt;&lt;
campo5&gt;&lt;/campo5&gt;&lt;campo6&gt;90125&lt;/campo6&gt;&lt;campo7&gt;CA&lt;/campo7&gt;&lt;campo8&gt;
US&lt;/campo8&gt;&lt;campo9&gt;711 Some Street&lt;/campo9&gt;&lt;campo10&gt;&lt;/campo10&gt;&lt;
campo11&gt;555-123-1234&lt;/campo11&gt;&lt;/subscriber&gt;&lt;/subscribers&gt;
      </ws:xmlDoc>
      <ws:groupsIDs></ws:groupsIDs>
      <ws:importType>3</ws:importType>
      <ws:mobileInputType>1</ws:mobileInputType>
      <ws:ConfirmEmail>false</ws:ConfirmEmail>
      <ws:forceOptIn>true</ws:forceOptIn>
      <ws:asPending>false</ws:asPending>
      <ws:asOptOut>false</ws:asOptOut>
      <ws:replaceGroups>false</ws:replaceGroups>
    </ws:StartImportProcesses>
  </soapenv:Body>
</soapenv:Envelope>
```

An example SOAP response returned by `StartImportProcess` method. Response may contain more than one `process` element, depending on how many lists were specified as input parameters

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <StartImportProcessesResponse xmlns="http://ws.mailupnet.it/">
      <StartImportProcessesResult><![CDATA[
        <?xml version="1.0" encoding="windows-1252" ?>
          <mailupMessage>
            <mailupBody>
              <ReturnCode>0</ReturnCode>
              <processes>
                <process>
                  <processID>69</processID>
                  <listID>1</listID>
                  <ReturnCode>0</ReturnCode>
                </process>
                <process>
                  <processID>70</processID>
                  <listID>2</listID>
                  <ReturnCode>0</ReturnCode>
                </process>
              </processes>
            </mailupBody>
          </mailupMessage>]]>
      </StartImportProcessesResult>
    </StartImportProcessesResponse>
  </soap:Body>
</soap:Envelope>
```

Error codes

Below is a list of the `ReturnCode` values, and what they mean when an error is encountered.

Error code	Description
-200	unrecognized error (it is likely that a mandatory parameter is missing)
-400	unrecognized error
-401	xmlDoc is empty
-402	convert xml to csv failed
-403	create new import process failed
-410	can not create confirmation email
-450	listsIDs and listsGUIDs must contain the same number of elements
-600	unrecognized error
-601	an import process is already running for the list
-602	an import process is already running for a different list
-603	error checking process status
-604	error starting the process job

Code Examples

The code examples below are provided on an "as is" basis, without any warranty of any kind. MailUp shall not be held liable for any direct, indirect or consequential damages or costs of any type arising out of any action taken by you or other related to the example code. Please contact us if you are interested in submitting examples for programming languages that are not already included below.

Ruby

```
# Refer to https://mailup.atlassian.net/wiki/display/mailupapi/MailUp+RubyGem for gem information.

require 'mail up'
m = MailUp::Import.new('username', 'password', 'console_url')
m.start_import_processes(
  listsIDs: "1;2;3",
  listsGUIDs: "abc123;def456;ghi789",
  xmlDoc: xml_data_string,
  groupsIDs: '',
  importType: 3,
  mobileInputType: 1
)
# => <?xml version=\"1.0\" encoding=\"windows-1252\" ?><mailupMessage><mailupBody><ReturnCode>0<
/ReturnCode><processes>...</processes></mailupBody></mailupMessage>
```

PHP

```php
<?php

class MailUpWsImport {
  protected $ns = "http://ws.mailupnet.it/";
  //replace <console host name> with the host name of your console
  protected $WSDLUrl = "http://<console host name>/services/WSMailUpImport.asmx?WSDL";
  protected $headers = array("User" => "user", "Password" => "password");
  protected $rCode;
  private $soapClient;
  private $xmlResponse;
  protected $domResult;

  function __construct() {
    $this->header = new SOAPHeader($this->ns, "Authentication", $this->headers);
    $this->soapClient = new SoapClient($this->WSDLUrl,array("trace"      => 1,"exceptions" => 0));
    $this->soapClient->__setSoapHeaders($this->header);
  }

  function __destruct() {
    unset($this->soapClient);
  }

  //...

  public function startImportProcesses($processData) {
    try {
      echo $processData."<br/><br/>";
      $this->soapClient->StartImportProcesses($processData);
    } catch (SoapFault $soapFault) {
      var_dump($soapFault);
    }
  }

  //...

}

$WsImport = new MailUpWsImport();
$xmlData = <<<EOT
<subscribers><subscriber email="test@example.com" Prefix="+39" Number="3351234567" Name="Test">
<campo1>Luigi</campo1><campo2>Rossi</campo2><campo3>Rossi consulting</campo3><campo4>Parma2<
/campo4><campo5>PR</campo5>
<campo6>43102</campo6><campo7></campo7><campo8>ITA</campo8><campo9>Via Garibaldi, 1<
/campo9><campo10>0521123456</campo10>
<campo11>0521123457</campo11></subscriber></subscribers>
EOT;

$startImportProcessData = array(
  "listsIDs" => "4",
  "listsGUIDs" => "3c5cb08c-f0b5-4bd6-9c2a-b687ecdac8b4",
  "xmlDoc" => $xmlData,
  "groupsIDs" => "22",
  "importType" => "3",
  "mobileInputType" => "2",
  "asPending" => '0',
  "ConfirmEmail" => "0",
  "asOptOut" => "0",
  "forceOptIn" => "0",
  "replaceGroups" => "0"
);

$WsImport->startImportProcesses($startImportProcessData);

?>
```

C#

```
WSMailUpImport toTest = new WSMailUpImport();

Authentication auth = new Authentication();
auth.User = tbLoginUserImport.Text;
auth.Password = tbLoginPasswordImport.Text;
auth.encType = string.Empty;

toTest.AuthenticationValue = auth;

string xmlString = "<subscribers><subscriber email=\"mike@example.com\" Prefix=\"\" Number=\"\" Name=\"\"
><campo1>Mike</campo1><campo2>Brown</campo2><campo3>Example Company</campo3><campo4>Los Angeles<
/campo4><campo5></campo5><campo6>90125</campo6><campo7>CA</campo7><campo8>US</campo8><campo9>555 Some
Street</campo9><campo10></campo10><campo11>555-123-1234</campo11></subscriber></subscribers>";

string myListId = "8";
string myListGuid = "CC9C4CBD-567B-4248-B56F-7C8364F11C5";
string myGroupsIDs = "45";
int importType = 3;
int mobileInputType = 2;
bool asPending =  false;
bool ConfirmEmail =  false;
bool asOptOut =  false;
bool forceOptIn =  false;
bool replaceGroups =  false;
string retVal = toTest.StartImportProcesses(myListId ,myListGuid, xmlString, myGroupsIDs, importType,
mobileInputType, asPending , ConfirmEmail, asOptOut, forceOptIn, replaceGroups);
```