



Campaign Commander™ Email & Mobile Edition Cloud Service APIs

Document name	Transactional Messaging Trigger API Guide
Service	Transactional Messaging Service for Triggering Email Messages
Protocol	SOAP & REST over HTTP
Version	10.8
Last updated on	May 20, 2013

Table of Contents

Table of Contents	2
Introduction	4
About This Document	4
About Campaign Commander™ Email & Mobile Edition APIs	4
Feedback	4
Support Options	4
Training Options	4
Useful Links	5
Disclaimer	5
Overview of the Transactional Messaging Trigger API	6
Introducing Transactional Messaging Trigger APIs	7
Module Overview	7
How it Works	8
Getting Started with Integration	10
Prerequisites	10
Quick Start	10
Integration Using Transactional Messaging APIs	11
URL Encoding Considerations (for HTTP GET methods only)	13
Security	14
Transactional Messaging Platform Big Picture	14
sendObject	15
SOAP Example	16
sendObjects	18
SOAP Example	19
sendObjectsWithFullStatus	21
SOAP Example	22
getSendRequestById	25
SOAP Example	25
Parameter List (for all call methods)	26
Synchronization Between Your Campaign Commander™ Email & Mobile Edition Account and the Transactional Messaging Platform	28
Personalization Grammar Reference	29
FAQ	31
Push Content - [EMV CONTENT]	31
Push Field - [EMV DYN]	32
Promotion & Ad - [EMV INCLUDE]	32
Synchronization – sync data between Marketing campaigns & Transactional Messaging	33
API Trigger Calls	33
Sample Requests	34
XML/HTTP POST	34

SOAP Example Request	35
Sample Template	36
Reference	37

Introduction

About This Document

This document is a reference document for using Campaign Commander™ Email & Mobile Edition APIs. It does not explain the purpose or functions of Campaign Commander™ Email & Mobile Edition features. For information on these features, please consult the *Campaign Commander™ Online Help* or the *Campaign Commander™ Email & Mobile Edition User Guide*.

This document is intended for developers and project managers.

About Campaign Commander™ Email & Mobile Edition 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 Campaign Commander™ Email & Mobile Edition 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 Campaign Commander™ Email & Mobile Edition campaign to targeted recipients.

Feedback

The Transactional Messaging Trigger API Guide is constantly being enhanced to provide you with more and more information on using Campaign Commander™ Email & Mobile Edition API methods.

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

We look forward to hearing from you!

Support Options

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

Training Options

Emailvision 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 Emailvision 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 Emailvision Academy website:

<http://www.emailvisionacademy.co.uk>

Useful Links

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

Disclaimer

While the information contained in this publication is believed to be true and accurate, Emailvision 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 Emailvision.

Overview of the Transactional Messaging Trigger API

The Transactional Messaging Trigger API allows you to set up the triggers for sending your Transactional Messages.

For further information on Transactional Messaging, please consult the *Campaign Commander™ Email & Mobile Edition User Guide* or *Campaign Commander™ Online Help*.

The following methods are available:

Method	Description
sendObject	This method is used to send a Transactional Message to an email address. The response indicates whether the send was successful.
sendObjects	This method allows you to send a Transactional Message to multiple email addresses.
sendObjectsWithFullStatus	This method allows you to send a Transactional Message to multiple email addresses with a detailed status response.
getSendRequestById	This method retrieves a send request using its ID.

Introducing Transactional Messaging Trigger APIs

The Transactional Messaging Platform & Module (often referred to as Notification Messaging Platform or NMP) allows you to configure emails to be automatically sent after an event, which serves as a trigger for the email. These emails can be:

- Transactional messages:

Example: A reseller wants to send order confirmations with Campaign Commander™ Email & Mobile Edition Transactional Messaging Platform. When someone orders a DVD, a web service triggers a request to CCMD NMP that will send the order confirmation to the final buyer.

- Social Networking messages:

Example: A social network wants to send all their notifications with Campaign Commander™ Email & Mobile Edition Transactional Messaging Platform. When a member uploads new photos on the social network, a web service triggers the social network website backend system to send a request to CCMD NMP triggering the broadcast of the notification to all member friends.

Module Overview

In brief, there are several types of messages that can be handled with the new Transactional Messaging Platform & Module.

Transactional E-mails:

- Order and service confirmations
- Shipment notifications
- Reservation confirmations and e-tickets
- "Available now" notifications
- Billing and payment notifications
- Cancellations, returns, refunds, rebates, and bonuses
- Information inquiry responses
- Government responses
- Product warranty information
- Periodic account balance notifications

Social Networking (+ Dating) E-mails:

- Welcome e-mails
- Profile update notifications
- Friend activity notifications (photos, pokes, etc.)
- Social network activity alerts
- Matching profile alerts
- Dating alerts
- Inbox alerts
- Viral marketing

- Job notifications
- “Lost password” confirmations

The Campaign Commander™ Transactional Messaging Trigger API set provides the following functionality and interfaces:

- The ability to trigger a “call to send” a Transactional Message
- Personalization capabilities:
 - Push Field ([EMV DYN] tag)
 - Push Content ([EMV CONTENT] tag, e.g. blocks of HTML code)
 - Promotion & Ad ([EMV INCLUDE] tag, i.e. bits of code or text managed within the Campaign Commander™ Email & Mobile Edition application)
- Status monitoring of a specific triggered message
- Functions to synchronize the marketing campaign data table (also known as the CCMD “MEMBER table)
- Scheduled message broadcasts
- High security standards using a tag set encrypt/decrypt method
- HTTPS interfaces for data security

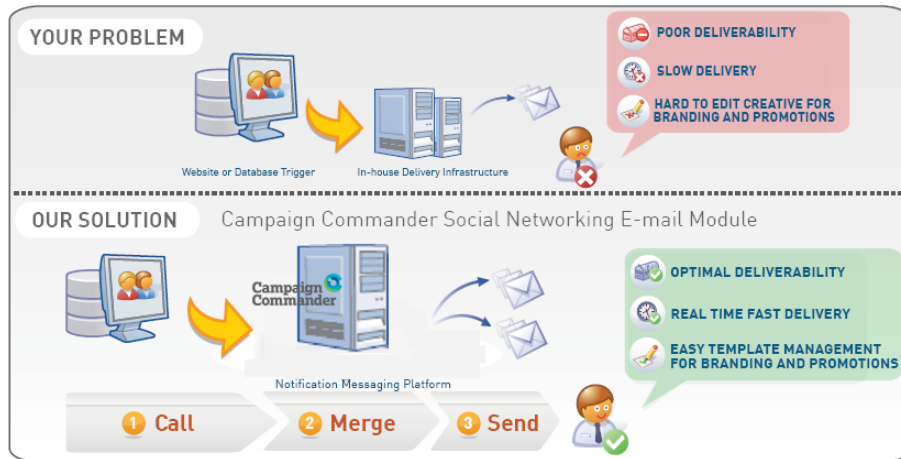
Transactional Messages are triggered in real time. Their purpose and time sensitivity are very different to standard marketing e-mail campaigns, like newsletters or promotional e-mails.

The Transactional Messaging Platform & Module thus provide:

- A high performance and robust dedicated platform including dedicated MTAs, trackers, and database servers
- A scalable platform with redundant servers and failover
- Highly dynamic personalization capabilities are pushed into the Transactional Messages using the different Trigger methods
- Easy integration with standard web methods (HTTP)
- Higher deliverability standards
- Real time & fast broadcast
- Live response reports
- Recipient history search and tracking features

How it Works

A quick overview of the Transactional Messaging “call to send” cycle:



Getting Started with Integration

This section describes all the steps involved in getting started with integrating Transactional Messaging (also referred to as Notification Messaging) with your website.

Prerequisites

To access Campaign Commander™ Email & Mobile Edition's NMP 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 Campaign Commander™ Email & Mobile Edition account with the Notification Messages feature enabled

Quick Start

The process for interfacing your website, CRM, or any other internal system with the Transactional Messaging APIs is quite straightforward. All interfacing and message creation can be setup within 5-10 business days depending on your teams' resources and expertise, in the following order:

Step 1: Build your Template(s) in Campaign Commander™ Email & Mobile Edition

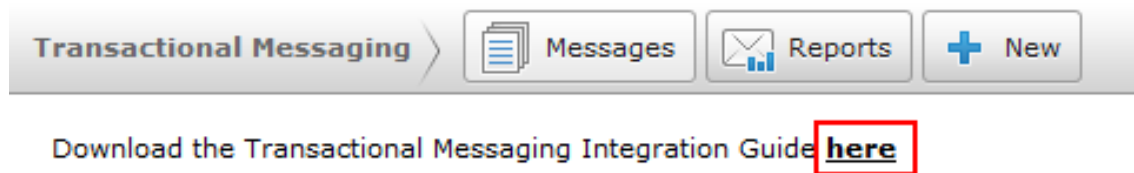
1. Contact your Emailvision Account Manager to
 - activate the Transactional Messaging feature for you in order to get started
 - set up the deliverability
2. Create or import the creative in Campaign Commander™ Email & Mobile Edition.
3. Create one or more Templates in Campaign Commander™ Email & Mobile Edition in the **Transactional Messaging** section.
4. Set up the Template using personalization features.
5. Preview and test the message internally.
6. Provide your IT team with the API documentation and the Unique Identifier and Security Tags output by Campaign Commander™ Email & Mobile Edition for the created Templates. They will need to use the unique identifier and the security tag in the API calls.



Template Summary	
Your Template was saved Successfully	
Integration Details	
Unique Identifier	C4DC168080000619
Security tag	EdX7CqkmjTHG8SA9MKJP2wqkLDgJHavL-jDde6k2XsHQKt4

Step 2: IT integration and testing

1. The webmaster and/or development team(s) can take over and begin the integration using the Transactional Messaging Trigger APIs (documentation available when clicking the link provided in Campaign Commander™ Email & Mobile Edition).



- The IT team codes the calls from their current system once and for all.

Step 3: Test the message and put it into production

Integration Using Transactional Messaging APIs

The first step in getting started with web services is to configure the range of remote servers that will access this module. Keep in mind that your system will require coding in order to remotely call the Transactional Messaging Platform & Service. Your system remains the remote control and trigger of a message.

Webmasters and developers should be able to interface with the Transactional Messaging API with any programming language that uses standard HTTP calls.

List of APIs that are available:

- RESTful API
 - HTTP Get Query String
 - HTTP GET Path Info
- SOAP

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.

This API protocol is 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://api.notificationmessaging.com/NMSREST?random={IntegrationTag}&encrypt={SecurityTag}
&email={Email}&senddate={SendDate}&uidkey={columnName}&stype={synchroType}&dyn={syncKey:
value|field:value|field:value}
```

HTTP GET (Query String)**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.

Sample Call 1 (with Synchro)

```
http://api.notificationmessaging.com/NMSREST?random=4A776E3602000078&encrypt=BdX7CqkmjTHtxWEKB5QK6MzXKkx6HK3E8guM&email=johnblum@flowerpowershop.biz&senddate=2008%2D12%2D12%2023%3A30%3A00&uidkey=email&stype=UPDATE&dyn=email:johnblum@flowerpowershop.biz|firstname:john|lastname:blum
```

Sample Call 2 (without Synchro)

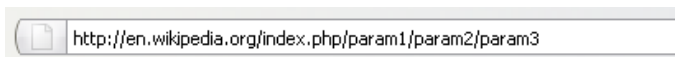
```
http://api.notificationmessaging.com/NMSREST?random=4A776E3602000078&encrypt=BdX7CqkmjTHtxWEKB5QK6MzXKkx6HK3E8guM&email=johnblum@flowerpowershop.biz&senddate=2008%2D12%2D12%2023%3A30%3A00&uidkey=&stype=NOTHING&dyn=firstname:john|lastname:blum
```

- 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://api.notificationmessaging.com/NMSREST/{IntegrationTag}/{SecurityTag}/{Email}/{SendDate}/{uidKey}/{SynchroType}/{syncKey:value|field:value|field:value}
```

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.

Example Call

```
http://api.notificationmessaging.com/NMSREST/4A776E3602000078/BdX7CqkmjTHtxWEKB5QK6MzXKkx6HK3E8guM/johnblum@flowerpowershop.biz/2008%2D12%2D12%2023%3A30%3A00/clienturn/UPDATE/clienturn:2R34T|firstname:john|lastname:blum
```

XML/HTTP API

The HTTP/XML consists in posting an XML file through HTTP POST. It is also the only batching method available. Using an HTTP Post of an XML file, clients can send multiple sending requests in one go. The XML file structure is shown in [Sample Requests](#).

Basically, HTTP/XML allows you to:

- Submit one or more new Notification Message request(s). For performance reasons, it is highly recommended.
- Obtain full access to all personalization features:
 - Push Field **[EMV DYN]** tag
 - Push Content **[EMV CONTENT]** tag
 - Promotion & Ad **[EMV INCLUDE]** tag
- API call summary:

XML/HTTP
Submission URL http://api.notificationmessaging.com/NMSXML
Parameters & associated values <ul style="list-style-type: none"> • 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.
Example Call See the XML sample file in Sample Requests for more details

SOAP API

- WSDL

WSDL URL
http://api.notificationmessaging.com/NMSOAP/NotificationService?wsdl

- Methods

sendObject sendObjects sendObjectsWithFullStatus
--

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.

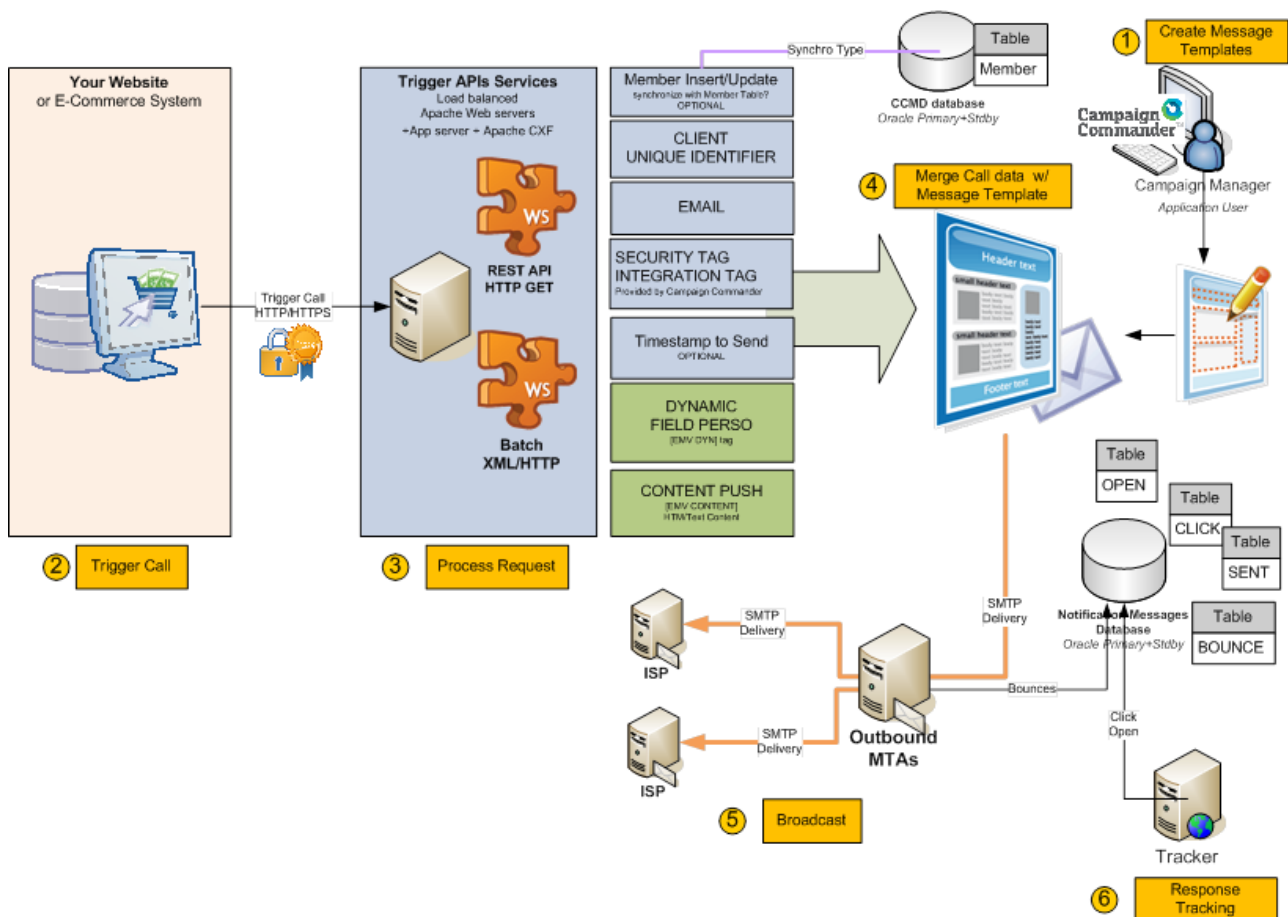
- The forward slash (/) is encoded as %2F.
- All other characters are encoded as %FF hex representation with any non-ASCII characters first encoded as UTF-8 (or other specified encoding).
- To encode as RFC 1738, use the + sign to replace spaces.

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, Campaign Commander™ Email & Mobile Edition 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.

Transactional Messaging Platform Big Picture



sendObject

This method is used to send a Transactional Message to an email address. The response indicates whether the send was successful.

WSDL location

<http://api.notificationmessaging.com/NMSOAP/NotificationService?wsdl>

Input parameters	Description	Output parameters	Description
Required parameters			
email	The email address to which you wish to send the Transactional Message	return	The status of the send
encrypt	The encrypt value provided in the interface		
notificationId	The ID of the Template		
random	The random value provided for the Template		
senddate	The time you wish to send the Transactional Message. If you put a past time, it will be sent immediately.		
synchrotype	The type of synchronization that should be carried out: <ul style="list-style-type: none"> • NOTHING: The member table will not be synchronized. • INSERT: The new profile will be inserted with all the associated DYN data in the member table. • UPDATE: The profile and all the associated DYN data will be updated in the member table. • INSERT_UPDATE: The system will try to update the profile in the member table. If the update fails, it tries to insert the profile with the associated DYN data into the member table. 		
uidkey	The key you wish to update, normally its email		
Content Parameters			
content	The content parameter envelope		
key	The key stated in the Template		
value	The value of the content key		
Dynamic Personalization Parameters			
dyn	The personalization parameter envelope		
key	The key stated in the Template		
value	The value of the dyn key		

Error messages
You must fill in the id parameter.
You must fill in the random parameter
You must fill in the encrypt parameter
You must fill in the email parameter
You must fill in the senddate parameter
You must fill in the uidkey parameter

Error messages
You must fill in the stype parameter
The email address {0} is malformed!!
Error while parsing dyn parameter: {0}.
Error while parsing sendDate parameter: {0}.
Error while parsing encrypt parameter: {0}
Query random: {0} doesn't match with notification random: {1} !!!
You don't have enough rights to use this service
The status not exist
The senddate doesn't exist or is malformed !
Job done or the sendRequest not exist
Unable to retrieve the notification
Unable to retrieve the client for this notification
Bad authentication!!! Random parameter is wrong {0}
Wrong encrypt parameter!
An error occured on the server

SOAP Example

Input

```
<soapenv:Body>
  <api:sendObject>
    <arg0>
      <content>
        <entry>
          <key>1</key>
          <value>
            <![CDATA[
              <table width="600">
                <tr>
                  <td>
                    <font size="2" face="Arial">Our powerful algorithms already
found a matching profile that matches your criteria:
                    <br>Celina72&nbsp;</font>
                    
                  </td>]]></value>
          </entry>
        </content>
        <dyn>
          <entry>
            <key>FIRSTNAME</key>
            <value>john</value>
          </entry>
        </dyn>
        <email>johnblum@flowerpower.com</email>
        <encrypt>BdX7CqkmjSivyBgIcZoN4sPVLkx7FaXGiwsO</encrypt>
        <notificationId>6464</notificationId>
        <random>985A8B992601985A</random>
        <senddate>2008-12-12T00:00:00</senddate>
        <synchrotype>NOTHING</synchrotype>
        <uidkey>EMAIL</uidkey>
```



```
        </arg0>
      </api:sendObject>
    </soapenv:Body>
  </soapenv:Envelope>
```

Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:sendObjectResponse xmlns:n-
s2="http://api.service.nsapi.emailvision.com/">
      <return>SendRequest has been successfully saved!</return>
    </ns2:sendObjectResponse>
  </soap:Body>
</soap:Envelope>
```

sendObjects

This method allows you to send a Transactional Message to multiple email addresses.

WSDL location

<http://api.notificationmessaging.com/NMSOAP/NotificationService?wsdl>

Input parameters		Output parameters	
Required parameters	Description		Description
email	The email address to which you wish to send the Transactional Message	return	The status of the send
encrypt	The encrypt value provided in the interface		
notificationId	The ID of the Template		
random	The random value provided for the Template		
senddate	The time you wish to send the Transactional Message. If you put a past time, it will be sent immediately.		
sychrotype	The type of synchronization that should be carried out: <ul style="list-style-type: none"> • NOTHING: The member table will not be synchronized. • INSERT: The new profile will be inserted with all the associated DYN data in the member table. • UPDATE: The profile and all the associated DYN data will be updated in the member table. • INSERT_UPDATE: The system will try to update the profile in the member table. If the update fails, it tries to insert the profile with the associated DYN data into the member table. 		
uidkey	The key you wish to update, normally its email		
Content Parameters			
content	The content parameter envelope		
key	The key stated in the Template		
value	The value of the content key		
Dynamic Personalization Parameters			
dyn	The personalization parameter envelope		
key	The key stated in the Template		
value	The value of the dyn key		

Error messages
You must fill in the id parameter.
You must fill in the random parameter
You must fill in the encrypt parameter

Error messages
You must fill in the email parameter
You must fill in the senddate parameter
You must fill in the uidkey parameter
You must fill in the stype parameter
The email address {0} is malformed!!
Error while parsing dyn parameter: {0}.
Error while parsing sendDate parameter: {0}.
Error while parsing encrypt parameter: {0}
Query random: {0} doesn't match with notification random: {1} !!!
You don't have enough rights to use this service
The status not exist
The senddate doesn't exist or is malformed !
Job done or the sendRequest not exist
Unable to retrieve the notification
Unable to retrieve the client for this notification
Bad authentication!!! Random parameter is wrong {0}
Wrong encrypt 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.nsapi.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:sendObjects>
      <arg0>
        <sendrequest>
          <content>
            <entry>
              <key>1</key>
              <value>
                <![CDATA[
                  <table width="600">
                    <tr>
                      <td>
                        <font size="2" face="Arial">Our powerful algorithms
already found a matching profile that matches your criteria:
                        <br>Celina72&nbsp;</font>
                        
                      </td>]]></value>
            </entry>
          </content>
        </sendrequest>
      </arg0>
    </api:sendObjects>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <email>johnblum@flowerpower.com</email>
        <encrypt>BdX7CqkmjSivyBgIcZoN4sPVLkx7FaXGiwsO</encrypt>
        <notificationId>6464</notificationId>
        <random>985A8B992601985A</random>
        <senddate>2008-12-12T00:00:00</senddate>
        <synchronotype>NOTHING</synchronotype>
        <uidkey>EMAIL</uidkey>
    </sendrequest>
</sendrequest>
<content>
    <entry>
        <key>1</key>
        <value>
            <![CDATA[
                <table width="600">
                    <tr>
                        <td>
                            <font size="2" face="Arial">Our powerful
algorithms already found a matching profile that matches your criteria:
                            <br>Celina72</font>
                            
                        </td>]]></value>
        </entry>
    </content>
    <dyn>
        <entry>
            <key>FIRSTNAME</key>
            <value>john</value>
        </entry>
    </dyn>
    <email>johnsmith@emailvision.com</email>
    <encrypt>BdX7CqkmjSivyBgIcZoN4sPVLkx7FaXGiwsO</encrypt>
    <notificationId>6464</notificationId>
    <random>985A8B992601985A</random>
    <senddate>2008-12-12T00:00:00</senddate>
    <synchronotype>NOTHING</synchronotype>
    <uidkey>EMAIL</uidkey>
</sendrequest>
</arg0>
</api:sendObjects>
</soapenv:Body>
</soapenv:Envelope>

```

Output

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:sendObjectsResponse xmlns:n-
s2="http://api.service.nsapi.emailvision.com/">
      <return/>
    </ns2:sendObjectsResponse>
  </soap:Body>
</soap:Envelope>

```

sendObjectsWithFullStatus

This method allows you to send a Transactional Message to multiple email addresses with a detailed status response.

WSDL location

<http://api.notificationmessaging.com/NMSOAP/NotificationService?wsdl>

Input parameters Required parameters	Description	Output parameters	Description
email	The email address to which you wish to send the Transactional Message	return	The status of the send
encrypt	The encrypt value provided in the interface		
notificationId	The ID of the Template		
random	The random value provided for the Template		
sendDate	The time you wish to send the Transactional Message. If you put a past time, it will be sent immediately.		
synchrotype	The type of synchronization that should be carried out: <ul style="list-style-type: none"> • NOTHING: The member table will not be synchronized. • INSERT: The new profile will be inserted with all the associated DYN data in the member table. • UPDATE: The profile and all the associated DYN data will be updated in the member table. • INSERT_UPDATE: The system will try to update the profile in the member table. If the update fails, it tries to insert the profile with the associated DYN data into the member table. 		
uidkey	The key you wish to update, normally its email		
Content Parameters			
content	The content parameter envelope		
key	The key stated in the Template		
value	The value of the content key		
Dynamic Personalization Parameters			
dyn	The personalization parameter envelope		
key	The key stated in the Template		
value	The value of the dyn key		

Error messages
You must fill in the id parameter.
You must fill in the random parameter
You must fill in the encrypt parameter
You must fill in the email parameter
You must fill in the senddate parameter
You must fill in the uidkey parameter

Error messages
You must fill in the stype parameter
The email address {0} is malformed!!
Error while parsing dyn parameter: {0}.
Error while parsing sendDate parameter: {0}.
Error while parsing encrypt parameter: {0}
Query random: {0} doesn't match with notification random: {1} !!!
You don't have enough rights to use this service
The status not exist
The senddate doesn't exist or is malformed !
Job done or the sendRequest not exist
Unable to retrieve the notification
Unable to retrieve the client for this notification
Bad authentication!!! Random parameter is wrong {0}
Wrong encrypt parameter!
An error occured on the server

SOAP Example

Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.service.nsapi.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:sendObjectsWithFullStatus>
      <arg0>
        <sendrequest>
          <content>
            <entry>
              <key>1</key>
              <value>
                <![CDATA[<table width="600">
                  <tr>
                    <td>
                      <font size="2" face="Arial">Our powerful algorithms already
profile that matches your criteria:
                      <br>Celina72 </font>
                      ]]>
                </value>
              </entry>
            </content>
            <dyn>
              <entry>
                <key>FIRSTNAME</key>
                <value>john</value>
              </entry>
            </dyn>
            <email>johnblum@flowerpower.com</email>
            <encrypt>BdX7CqkmjSivyBgIcZoN4sPVLkx7FaXGiwsO</encrypt>
            <notificationId>6464</notificationId>
            <random>985A8B992601985A</random>
```

```

    <sendDate>2008-12-12T00:00:00</sendDate>
    <synchrotype>NOTHING</synchrotype>
    <uidkey>EMAIL</uidkey>
  </sendrequest>
  <sendrequest>
    <content>
      <entry>
        <key>1</key>
        <value>
          <![CDATA[<table width="600">
            <tr>
              <td>
                <font size="2" face="Arial">Our powerful algorithms already
profile that matches your criteria:
                <br>Celina72 </font>
                ]]>
            </td>
          </value>
        </entry>
      </content>
      <dyn>
        <entry>
          <key>FIRSTNAME</key>
          <value>john</value>
        </entry>
      </dyn>
      <email>johnsmith@emailvision.com</email>
      <encrypt>BdX7CqkmjSivyBgIcZoN4sPVLkx7FaXGiws0</encrypt>
      <notificationId>6464</notificationId>
      <random>985A8B992601985A</random>
      <sendDate>2008-12-12T00:00:00</sendDate>
      <synchrotype>NOTHING</synchrotype>
      <uidkey>EMAIL</uidkey>
    </sendrequest>
  </arg0>
</api:sendObjectsWithFullStatus>
</soapenv:Body>
</soapenv:Envelope>

```

Output

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:sendObjectsWithFullStatusResponse xmlns:n-
s2="http://api.service.nsapi.emailvision.com/">
      <return>
        <element responseStatus="success" email="johnblum@flowerpower.com">
          <result xsi:type="ns4:string" xmlns:x-
si="http://www.w3.org/2001/XMLSchema-instance" xmlns:n-
s4="http://www.w3.org/2001/XMLSchema">SendRequest has been successfully
saved</result>
        </element>
        <element responseStatus="success" email="pishty@emailvision.com">
          <result xsi:type="ns4:string" xmlns:x-
si="http://www.w3.org/2001/XMLSchema-instance" xmlns:n-
s4="http://www.w3.org/2001/XMLSchema">SendRequest has been successfully
saved</result>
        </element>
      </return>
    </ns2:sendObjectsWithFullStatusResponse>
  </soap:Body>
</soap:Envelope>

```

```
</ns2:sendObjectsWithFullStatusResponse>  
</soap:Body>  
</soap:Envelope>
```


getSendRequestById

This method retrieves a send request using its ID.

WSDL location

<http://api.notificationmessaging.com/NMSOAP/NotificationService?wsdl>

Input parameters	Description	Output parameters	Description
Required parameters			
id	The id of the send request	return	The send request

Error messages
You must fill in the id parameter.
The status not exist
Job done or the sendRequest not exist
An error occurred on the server

SOAP Example

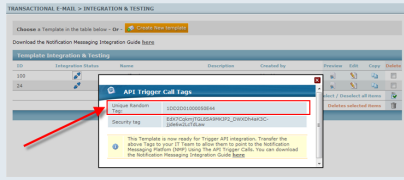
Input

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:api="http://api.service.nsapi.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:getSendRequestById>
      <arg0>65</arg0>
    </api:getSendRequestById>
  </soapenv:Body>
</soapenv:Envelope>
```

Output

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getSendRequestByIdResponse xmlns:n-
s2="http://api.service.nsapi.emailvision.com/">
      <return>
        <id>0</id>
        <notificationId>0</notificationId>
        <status>Job done or the sendRequest not exist</status>
      </return>
    </ns2:getSendRequestByIdResponse>
  </soap:Body>
</soap:Envelope>
```

Parameter List (for all call methods)

Parameter Required parameters	Description	Values
random	Integration Tag The random key used to match with the Security Tag	Retrieved from Campaign Commander™ Email & Mobile Edition Transactional Messaging Template creation interface. Each Template has its own unique random integration tag. 
encrypt	Security Tag The encryption key used to retrieve the message Template ID and match with the Integration Tag	Same as above. Each Template has its own unique encryption key.
email	The recipient's email address to which you want to send the message	Alphanumeric value A syntax check is performed on the email address before sending.
senddate	The date when the message has to be sent (TimeZone GMT+1). Two options: <ul style="list-style-type: none"> If the date is in the future, the system will send it at that date/time If the date is in the past, the message will be sent immediately 	<u>XML/HTTP:</u> YYYY-MM-DDTHH:MM:SS <ul style="list-style-type: none"> YYYY is the year in 4 digits (1970) MM is the month in 2 digits (01) DD is the day in 2 digits (01) The "T" is used as a separator between date and time blocks E.g. 2008-12-24T23:30:00 <u>HTTP GET (Path Info and Query String):</u> YYYY%2DMM%2DDDD%20HH%3AMM%3ASS <ul style="list-style-type: none"> YYYY is the Year in 4 digits (1970) MM is the Month in 2 digits (01) DD is the Day in 2 digits (01) The %20 sign is a URL encoded space. The %2D sign is a URL encoded hyphen (-). The %3A sign is a URL encoded colon (:). E.g. 2008%2D12%2D12%2023%3A30%3A00

Parameter Required parameters	Description	Values
uidkey	The name of the column used to match with the member table when synchrotype is ON. It will match the DYN parameters that have the same name to retrieve the associated value	Alphanumeric values default: EMAIL
synchrotype (XML) stype (GET)	The type of synchronization to perform for each SendRequest between the Transactional Messaging platform and the Campaign Commander™ member table.	These values are case sensitive. <ul style="list-style-type: none"> • NOTHING: The member table will not be synchronized. • INSERT: The new profile will be inserted with all the associated DYN data in the member table. • UPDATE: The profile and all the associated DYN data will be updated in the member table. • INSERT_UPDATE: The system will try to update the profile in the member table. If the update fails, it tries to insert the profile with the associated DYN data into the member table.
dyn	Dynamic field personalization container for: <ul style="list-style-type: none"> • Simple personalization • DYN parameter/value pairs do not necessarily need to be used in the Template and can be included in the API call only for updating/inserting into the member table • If synchrotype ≠ NOTHING, then all DYN fields will be matched to the member table columns for inserting/updating the profile. 	<ul style="list-style-type: none"> • HTTP GET (PI or QS) &dyn=key:value field:value field:value • XML/HTTP <dyn> <entry> <key>fieldname</key> <value>thevalue</value> </entry> </dyn>
content	Portion of pre-formatted HTML or text blocks (see URL Encoding Considerations (for HTTP GET methods only) on page 13)	It can contain: <ul style="list-style-type: none"> • HTML code: The code has to be enclosed between the following tags: <![CDATA[YOUR_CODE_HERE]]> • Text only

Synchronization Between Your Campaign Commander™ Email & Mobile Edition Account and the Transactional Messaging Platform

You can synchronize the Transactional Messaging profile data coming from the Transactional Message requests with your account members (subscribers, clients). Transactional Messaging allows you to decide within the Trigger call to add or update a profile with the marketing account members.

Example: Update the last date of purchase of a High Tech Newsletter subscriber who has purchased a product on your website and was sent a product order confirmation.

All fields that have to be updated or inserted must be included within the **<DYN>** section for the HTTP POST/XML and within the **&dyn=** parameters list for the HTTP GET.

Fields that need to be synchronized must be predefined and set up on the Transactional Messaging Platform. Please ask your Account Manager to add them at setup time.

You can update all types of fields. Just make sure that when updating numeric and date fields, you check the values to avoid synchronization errors.

For date fields, you can find the allowed formats below:

- MM/dd/yyyy HH:mm
- MM/dd/yyyy HH:mm:ss
- MMM dd yyyy HH:mma
- yyyy-MM-dd HH:mm:ss
- MM/dd/yyyy

Personalization Grammar Reference

Two personalization options are available in the Transactional Messaging API set when calling the APIs. Other possibilities are also available at message creation (Dynamic Content, conditional statements). These possibilities are described in the *Transactional Messaging Templates API Guide*.

Tag name	Purpose	Example
Push Field [EMV DYN] PARAMETER_ NAME [EMV /DYN]	<p>Standard field personalization within Templates. The value/pair must be pushed within every API call request. The name of the parameter (either in the HTTP GET or in the XML file) must:</p> <ul style="list-style-type: none"> • be alphanumeric and cannot contain any spaces, special characters, or decimals. • match the name used between the [EMV DYN] and [EMV /DYN] tags in the Template. <p>This field does not support any type of HTML tags. These tags can be filled with simple text, numbers, and special characters that are included in the variable. The encoding accepted by Campaign Commander™ is the following:</p> <ul style="list-style-type: none"> • General US & West European (ISO-8859-15) • Unicode (UTF-8) • Cyrillic (windows-1251) • Traditional Chinese (Big5) • Simplified Chinese (GB2312) • Korean (EUC-KR) • Japanese (Shift-JIS) • Eastern Europe (ISO-8859-2) • Turkish (ISO-8859-3) • Greek (ISO-8859-7) • Latvian, Estonian (windows-1257) <p>Note: The Transactional Messaging Template should have the same encoding as the DYN fields.</p>	<ul style="list-style-type: none"> • Template: "Dear [EMV DYN] FIRSTNAME [EMV /DYN]," • Using API Get QS call: http://....&firstname=simon

Tag name	Purpose	Example
Push Content [EMV CONTENT] ID [EMV /CONTENT]	<p>Push blocks of pre-formatted HTML or text content within the API call to use as personalization within the Template. Each block of content can be different per request or recipient (even in the same XML file). A call can include several content blocks.</p> <p>Personalize your messages with pre-calculated matching profile pictures and text pulled from your HTTP GET or in the XML file.</p> <p>This field can also be filled in same way as the Dyn tag, but this tag also supports any type of HTML tags. (To insert HTML into a CONTENT field you need to enclose the HTML code in a <![CDATA[]]> parameter).</p>	<ul style="list-style-type: none"> Template: "Your matching profiles:
[EMV CONTENT] 1 [EMV /CONTENT]" API call using XML/HTTP: <content> <entry> <key>1</key> <value><![CDATA[<table><tr><td>matching profile:
</td></tr></table>]></value> </entry> </content>

FAQ

Push Content - [EMV CONTENT]

Q: What is the [EMV CONTENT] personalization designed for?

A: This tag is designed for managing personalization. It is a container for pushing portions of HTML Content (like a HTML table for e.g.) or Text into an API call to the NMP.

E.g. A dating website could push matching profile content for each individual subscriber.

Q: Is [EMV CONTENT]ID[EMV /CONTENT] linked to one Template?

A: Yes, one Push Content ID is linked to one Template.

Q: What is the difference between [EMV DYN] and [EMV CONTENT]?

A: [EMV DYN] Push field is for simple "one field" personalization. [EMV CONTENT] is for pre-formatted blocks of HTML or Text.

Q: Can links be tracked in [EMV CONTENT]?

A: Yes, to track one or more links in a message, place the URLs within the URL tracking tags[EMV URL][EMV /URL]. In the NMP export, the following information for a clicked link is given:

- message ID
- member ID
- message name
- member email
- click date
- link name
- link type
- root of the link URL

Note: The root of the URL will be used as the reference for tracking behavior.

Example: If the following link is tracked:

`https://www.hereismyrooturl/ecom/gb.php?c=andtheparameter=1094184&cl=148641&ejc=2&o1=BABYBLUE-BT-BB&quantity=1`

then only the part of the URL up to the question mark will be exported:

`https://www.hereismyrooturl/ecom/gb.php?`

Thus, you would get the following information for a tracked link that was clicked in an example NMP export:

```
"nmMessageId": 45693, "memberUID": "59BCDC04000106AD", "name": "nmp_content_test_msg", "email": "xxx@xxx.com", "clickDate": "2012-08-13 00:00:00.0", "linkName": "DYNAMIC_TRACKING", "linkType": 8, "url": "https://www.hereismyrooturl/ecom/gb.php?"
```

Q: Can I use personalization in [EMV CONTENT] blocks?

A: No, it is not possible to use personalization tags within [EMV CONTENT] data, they would not be interpreted when merged into the Template.

Q: How do I secure transfer of special characters (Chinese or Russian...etc) as Push Content?

A: The XML file has to be encoded in UTF-8 (content coming from client in the API Trigger Call).

Then, the Template must also be encoded in the defined character when building the message, as below:

Encoding:	Unicode (UTF-8) 
-----------	---

Q: Can I use [EMV CONTENT] and [EMV DYN] tags within an [EMV INCLUDE] Dynamic Content Block?

A: Yes, you can embed these 2 personalization tags within a Dynamic Content Block [EMV INCLUDE], they'll be interpreted properly.

Q: Can I use [EMV CONTENT] to transfer the whole body of my HTML message?

A: Not recommended. [EMV CONTENT] is not meant to be used for pushing the whole message body, and the reasons are simple:

- The XML can become very heavy when it comes to batching many recipients in one XML file.
- You would have to create a Template in your account anyway, with just the [EMV TEXTPART] and [EMV HTMLPART] <html><head></head>XXXXX</body></html>.
- Links are not tracked.
- You cannot use personalization.

Q: What is the difference between [EMV CONTENT] and [EMV INCLUDE]?

A: [EMV INCLUDE] and [EMV CONTENT] have the same goal: handle blocks of Messages separately from the Template. The differences are:

- [EMV INCLUDE] are blocks managed in the Campaign Commander Application and can handle personalization & trackable links.
- [EMV CONTENT] are pushed in every API Trigger Call and can be different for every Trigger Call.

Push Field - [EMV DYN]

Q: What is the [EMV DYN] tag made for?

A: [EMV DYN] are parameters pushed through the API Trigger Call. They are to be used for "simple" personalization (key=value).

Q: If I use many times [EMV DYN] personalization in my Template, do I have to include it once or many times in the API call?

A: You just have to include the DYN once in the API Call and it will be merged at all locations in the Template.

Promotion & Ad – [EMV INCLUDE]

Q: What is the [EMV INCLUDE] tag designed for?

A: [EMV INCLUDE] are portions of HTML code or Text stored in the NMP database (exactly like MESSAGES). They are inserted in the message at point of send. [EMV INCLUDE] can be used in several messages. There is an interface in Campaign Commander™ that lets you create/modify/delete/save includes, under the Dynamic Content Block section of Transactional Messaging. If you modify an include content, all messages that contain this [EMV INCLUDE] will use the modified include in following notification.

Q: Can we use it also for normal messages?

A: Not yet. It is only available for Transactional Messaging Templates for now.

Synchronization – sync data between Marketing campaigns & Transactional Messaging

Q: What is the matching key used to synchronize with my Campaign Commander™ MEMBER table?

A: The matching column key is specified within the call by the uidkey parameter. uidkey contains the name of the table column (E.g. MY_UNIQUE_ID). The value associated with that matching key is defined in the DYN fields parameters list section using the same name as the parameter name.

E.g.

- HTTP GET Trigger Call: &dyn=MY_UNIQUE_ID:<value>
- XML/HTTP:

```
<uidkey>CLIENTURN</uidkey>

<dyn>

<entry>

<key>CLIENTURN</key>

<value>123456789</value>

</entry>

</dyn>
```

Q: Which data is updated?

A: All Push Fields included in the DYN section will update the profile based on the matching key.

API Trigger Calls

Q: Are all DYNamic field/value pairs in the API call required to be in my Template?

A: No, DYNamic parameters can only be used for synchronizing with your Campaign Commander™ MEMBER table and do not necessarily have to be present within the Template.

Q: If my XML file or my API call is not compliant, what happens?

A: For any exception or error during the processing of an API call, a 500 HTTP error code is then returned to the requester service. If all OK, a "OK" HTTP code 200 is returned

Sample Requests

XML/HTTP POST

```
<?xml version="1.0" encoding="UTF-8"?>
  <MultiSendRequest>
    <sendrequest>
      <dyn>
        <entry>
          <key>firstname</key>
          <value>john</value>
        </entry>
        <entry>
          <key>lastname</key>
          <value>blum</value>
        </entry>
        <entry>
          <key>email</key>
          <value>jblum@flowerpowershop.biz</value>
        </entry>
        <entry>
          <key>CLIENTURN</key>
          <value>2R34T</value>
        </entry>
      </dyn>
      <content>
        <entry>
          <key>1</key>
          <value>
            <![CDATA[
              <table width="600">
                <tr>
                  <td>
                    <font size="2" face="Arial">Our powerful algorithms already
found a matching profile that matches your criterias:
                    <br>Celina72&nbsp;</font>
                    
                  </td>]]></value>
          </entry>
          <entry>
            <key>2</key>
            <value>Our powerful algorithms already found a matching profile
that matches your criterias: Celina72</value>
          </entry>
        </content>
        <email>jblum@flowerpowershop.biz</email>
        <encrypt>BdX7CqkmjTHtxWEKB5QK6MzXKkx6HK3E8guM</encrypt>
        <random>4A776E3602000078</random>
        <senddate>2008-12-12T00:00:00</senddate>
        <synchronotype>UPDATE</synchronotype>
        <uidkey>CLIENTURN</uidkey>
      </sendrequest>
    </MultiSendRequest>
```

SOAP Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.service.nsapi.emailvision.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <api:sendObjectsWithFullStatus>
      <arg0>
        <sendrequest>
          <content>
            <entry>
              <key>1</key>
              <value>>
                <![CDATA[
                  <table width="600">
                    <tr>
                      <td>
                        <font size="2" face="Arial">Our powerful algorithms
already found a matching profile that matches your criterias:
                        <br>Celina72&nbsp;</font>
                        
                      </td>]]></value>
            </entry>
          </content>
          <dyn>
            <entry>
              <key>firstname</key>
              <value>john</value>
            </entry>
          </dyn>
          <email>jblum@flowerpowershop.biz</email>
          <encrypt>BdX7CqkmjTHtxWEKb5QK6MzXKkx6HK3E8guM</encrypt>
          <notificationId>1234</notificationId>
          <random>4A776E3602000078</random>
          <senddate>2008-12-12T00:00:00</senddate>
          <synchrotype>UPDATE</synchrotype>
          <uidkey>EMAIL</uidkey>
        </sendrequest>
        <sendrequest>
          <content>
            <entry>
              <key>1</key>
              <value>>
                <![CDATA[
                  <table width="600">
                    <tr>
                      <td>
                        <font size="2" face="Arial">Our powerful
algorithms already found a matching profile that matches your criterias:
                        <br>Celina72&nbsp;</font>
                        
                      </td>]]></value>
            </entry>
          </content>
          <dyn>
            <entry>
```

```

        <key>firstname</key>
        <value>David</value>
    </entry>
</dyn>
<email>dcoulon@flowerpowershop.biz</email>
<encrypt>BdX7CqkmjTHtxWEKB5QK6MzXKkx6HK3E8guM</encrypt>
<notificationId>1234</notificationId>
<random>4A776E3602000078</random>
<senddate>2008-12-12T00:00:00</senddate>
<synchrotype>UPDATE</synchrotype>
<uidkey>EMAIL</uidkey>
</sendrequest>
</arg0>
</api:sendObjectsWithFullStatus>
</soapenv:Body>
</soapenv:Envelope>

```

Sample Template

This Template matches the API calls above.

```

[EMV TEXTPART]
Dear [EMV DYN]FIRSTNAME[EMV /DYN] [EMV DYN]LASTNAME[EMV /DYN],
Thank you for subscribing to MyDate.
You are subscribed with the Email address:
[EMV DYN]EMAIL[EMV /DYN]
[EMV CONTENT]2[EMV /CONTENT]
Let us know if you have any question.
My Date Customer Service.
[EMV HTMLPART]
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>My Sample HTML</title>
  </head>
  <body>
    
    <table width="600" border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td>
          <font size="2" face="Arial">Dear [EMV DYN]FIRSTNAME[EMV /DYN] [EMV DYN]
LASTNAME[EMV /DYN], Thank you for subscribing to MyDate. You are subscribed with
the Email address: [EMV DYN]EMAIL[EMV /DYN]
          <font size="2" face="Arial">Let us know if you have any question.My
Date Customer Service.</font>
        </td>
      </tr>
    </table>[EMV CONTENT]1[EMV /CONTENT]
    <font size="2" face="Arial">Let us know if you have any question.My Date
Customer Service.</body>
</html>

```

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.