API Documentation

Release 2.35.1

Ubivox Developers

Apr 29, 2020

CONTENTS

Contents:

CHAPTER

ONE

ACCESSING THE API

1.1 Protocols

The API uses the XML-RPC protocol as defined in the XML-RPC Specification

All communication with the API are sent over HTTPS (SSL encrypted HTTP).

1.2 Location

Every Ubivox customer has their own hostname. From there you can access the XML-RPC API using HTTP POST on `/xmlrpc/`. An absulute url might look like this:

```
https://customer.clients.ubivox.com/xmlrpc/
```

1.3 Authentication

The API uses Basic HTTP authentication over an encrypted SSL HTTPS connection.

You create a password to use when authenticating with the API by clicking *Account* and *Users and Security*. Click the tab *API authentication* and click *New password*.

The new password will be shown to you once so make a note of it. The *API authentication* tab gives you an overview of all active passwords, when and from where they were last used and an option of deleting each password to remove access to the account.

We do not store your password in plaintext and thus we are not able to tell you your current password.

1.4 Throttling

In order for us to guarentee a great quality of service, we enforce a limit on API calls. The limit is based on the current hour of the clock.

When you reach the limit, the system will start replying with the XML-RPC fault code `9997` (*Too many calls (throttling)*).

1.5 Service windows

When Ubivox is not accessible due to service windows or unplanned downtime on the application servers, the system will reply with a `503 SERVICE UNAVAILABLE` to your requests.

In very rare occasions, the API load balancer may experience network problems and then you will receive a network timeout.

If either happens, then your request was *not* processed and you should repeat it at a later time.

All planned service windows will be announced ahead of time in newsletters as well on the status blog (which also contains information on unplanned downtime).

1.6 Request log

From your Ubivox web interface you can find a API request log. You will find it from *Account* then *API Requests*. This contains a searchable index of all your API requests from the past six months. This is a great tool for debugging your integration with Ubivox.

1.7 Debugging

During the development phase of your integration, it might be a help to dump requests and responses as they happen. We provide the option of having the raw requests sent to your e-mail address for your inspection.

If you want to activate this, you will find it in the Ubivox web interface from *Account* then *API*. Here you will be able to turn on *XML-RPC debugging*

CHAPTER

TWO

DEFINITIONS

2.1 Date/time formats and time zone

All date and time values can either be sent as strings or as the XML-RPC `dateTime.iso8601` type. If you choose to send it as strings, they must be sent with the following format:

Date: `YYYY-MM-DD` (E.g.: `2010-05-31`)

Date with time: `YYYY-MM-DD HH:MM:SS` (E.g.: `2010-05-31 18:42:53`)

Time: `HH:MM:SS` (E.g.: `18:42:53`)

If you choose to send you dates and times as the `dateTime.iso8601` type, you must conform to the XML-RPC specification.

All `dateTime.iso8601` timestamps are interpreted according to the account time zone.

There is also a special case to indicate the current time: Send the string `now`.

2.1.1 Time zone support

The timestamps you send to the API will be interpreted according to the time zone your account has been configured with. This can be configured from the Ubivox interface: *Account* and then *Contact details*.

Note: The current local time for the account (in the time zone, as configured on the contact details) can be looked up by calling *`mailer.server_time()`*.

Note: An array of the supported time zones can be fetched from the API using *`mailer.time_zones()`*.

2.2 Error codes

All error reporting is done through XML-RPC fault messages. The following error conditions can happen:

Error code	Error message
1001	Invalid e-mail address
1002	The user is not subscribed
1003	The user is already subscribed

Continued on next page

Table 1 – continued from previous page

Error code	Error message
1004	Invalid data field
1005	Invalid data field key
1006	Invalid data field type
1007	Invalid data field access
1008	The subscriber is not a test recipient
1009	Invalid subscriber
1010	Missing consent
1011	New subscriptions blocked for your account
2001	Invalid mailing list
2002	The course has already ended (no stage to advance to)
2003	Invalid stage.
2004	Opt-in not configured (on one or more lists).
2005	Invalid operation on archived list
3001	Invalid delivery
3002	Delivery not in edit state
3003	Delivery not in standby state
3004	Problem with delivery content
3005	Mail merging error
3006	Delivery not sent
3007	Invalid template
3008	Invalid template content
4001	Invalid import
4002	Too many invalid addresses in import file
4003	Another import already exists with this filename
4004	Invalid import filename
4005	Format error in import file
4006	Invalid export
4007	Export is not done processing yet
4008	Sender is not verified
5001	Invalid date/time format
5002	Invalid date format
5003	Invalid offset
5004	Invalid reference (only a-z, A-Z, 0-9, - and _ allowed)
5005	Invalid time zone
5006	Invalid language
5007	Invalid filename
5008	Invalid rule set
5009	Invalid timezone
5010	Invalid target
5011	Invalid webhook
5012	Invalid hook
5013	Invalid user
5014	Invalid subject
5015	Invalid percentage
5016	Invalid criteria variable
5017	Invalid criteria operator
5018	Invalid schedule
5019	Invalid encoding
5020	Invalid invoice

Continued on next page

Table 1 – continued from previous page

Error code	Error message
5021	Invalid or empty body
5022	Invalid sender or duplicates found
6001	Wrong number of parameters given
6002	No such method
7001	Invalid client
7002	FTP username taken
7003	Hostname taken
7004	Username taken
7005	Invalid user
7006	Account name taken
7007	Invalid account type
7008	Invalid account name
7009	Missing address field for customer
7010	Invalid method
7011	Cannot approve a customer that's not active
9992	Invalid multicall call struct
9993	Recursive multicall not allowed
9994	Unauthorized
9995	Account type upgrade needed (contact sales)
9996	Not approved for this call
9997	Too many calls (throttling)
9998	XML error (maybe not wellformed)
9999	Unknown error

In each method, error conditions that can occur is listed.

2.3 Subscriber data

All API methods that work with subscriber data do so by utilizing the XML-RPC `struct` type (see *Protocols*). The struct is a mapping from the data *key* to the data *value* for this particular subscriber.

2.3.1 Data fields with predefined options

When dealing with data fields with predefined options, you need to supply the *keys* of your options in a comma separated string as the *value* of the data field.

2.4 Supported hooks for webhooks

We currently support the following hooks/events for webhooks:

- *subscription*
- *unsubscription*
- *remove_subscription*
- *suspend_subscription*
- *activate_subscription*

- *login*
- *junk_delivery*
- *junk_email*
- *bounced_email*

For explanations, please refer to the Ubivox webhook interface located inside the Ubivox system from *Account* then *Webhooks*.

2.5 Stoplist causes

These are the different causes that may get a subscriber on the stoplist depending on your account configuration:

- *junkreport*: We received an automatic (or manual) junk report for the subscriber.
- *admin*: Administrative request.
- *unknown*: Unknown cause.
- *unsubscription*: Unsubscription (only if enabled by contacting Ubivox)
- *bounce*: Bounce (only if enabled by contacting Ubivox)

CHAPTER

THREE

API METHODS

Here are the methods supported by the API grouped by the primary entity they operate on.

3.1 `system` methods

`system.`**`listMethods`**`()`
List system methods. Part of XML-RPC introspection.

Returns Returns a list of method names.

Return type array

`system.`**`methodHelp`**`(`*`method`*`)`
Gets help about a method. Part of XML-RPC introspection.

Parameters **`method`** (*`string`*) – method name

Returns Returns a text describing the method.

Return type string

Raises `6002`: No such method

`system.`**`multicall`**`(`*`calls`*`)`
Run a batch/multicall XML-RPC request with multiple XML-RPC calls.

Parameters **`calls`** (*`array`*) – An array of structs with the XML-RPC calls according to the spec (with keys: `methodName`, `params`)

Returns An array of mixed types according to the return values of the calls

Return type array

Raises Any fault the embedded calls may raise.

Raises `9992`: Invalid multicall call struct

Raises `9993`: Recursive multicall not allowed

Note: This is an extension to the XMLRPC specification.

- Please refer to http://mirrors.talideon.com/articles/multicall.html for details.
- This implementation of `system.multicall` doesn't allow recursive calls. Those will return the `9993` fault code.
- The calls may be executed out of order, but the results will be in order (the calls are made concurrently on the server side).

- A maximum of 50 calls may be executed at once. Larger batches will raise `9992`.
- The above faults are all reported on their respective call in the `calls` array as reflected in the returned array. Exception: If the `calls` parameter is not of the array type, a "global" fault with code `9992` will be returned.

3.2 `subscription` methods

`mailer.`**`cancel_all_subscriptions`**(*email_address*)
Cancel all subscriptions for a given subscriber.

Parameters **`email_address`** (*`string`*) – The e-mail address of the subscriber.

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

`mailer.`**`cancel_subscription`**(*email_address*, *lists*)
Cancel a subscription for the lists specified.

Parameters

- **`email_address`** (*`string`*) – The address of the subscriber
- **`lists`** (*`array`*) – A list of mailinglist IDs

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

Raises `1002`: The user is not subscribed

Raises `2001`: Invalid mailing list

Note: If the user was not subscribed some of the lists, the NotSubscribed fault is raised.

`mailer.`**`course_force_stage`**(*subscriber*, *list*, *stage*)
Forces the subscriber to the specified stage on the course of the list.

Parameters

- **`subscriber`** (*`string`*) – The e-mail address of a subscriber.
- **`list`** (*`integer`*) – A mailing list ID.
- **`stage`** (*`integer`*) – An integer ID of a stage on the mailing list.

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

Raises `1002`: The user is not subscribed

Raises `2001`: Invalid mailing list

Raises `2003`: Invalid stage.

Raises 2005: Invalid operation on archived list

mailer.**course_next_stage**(*subscriber*, *list*)
Advances the subscriber to the next stage on the course of the list.

Parameters

- **subscriber** (*string*) – The e-mail address of a subscriber.
- **list** (*integer*) – A mailing list ID.

Returns True on success.

Return type boolean

Raises 1001: Invalid e-mail address

Raises 1002: The user is not subscribed

Raises 2001: Invalid mailing list

Raises 2002: The course has already ended (no stage to advance to)

Raises 2005: Invalid operation on archived list

mailer.**create_subscription**(*email_address*, *lists*, *optin*)
Create a subscription for the lists specified.

Parameters

- **email_address** (*string*) – The address of the subscriber
- **lists** (*array*) – A list of mailinglist IDs
- **optin** (*boolean*) – Should we send a optin e-mail

Returns True on success.

Return type boolean

Raises 1001: Invalid e-mail address

Raises 1003: The user is already subscribed

Raises 1011: New subscriptions blocked for your account

Raises 2001: Invalid mailing list

Raises 2004: Opt-in not configured (on one or more lists).

Raises 2005: Invalid operation on archived list

mailer.**create_subscription_error**(*email_address*, *optin*)
Get a detailed error description for any 1001 errors encountered during subscription processing.

Parameters

- **email_address** (*string*) – The address of the subscriber
- **optin** (*boolean*) – Does the new subscription require opt-in? (for stop list checking)

Returns A string describing the error. Will be empty for success.

Return type string

mailer.**create_subscription_with_data**(*email_address*, *lists*, *optin*, *data*)
Create a subscription for the lists specified with data from the data struct.

Parameters

- **email_address** (*string*) – The e-mail address of the subscriber
- **lists** (*array*) – A list of mailing list IDs
- **optin** (*boolean*) – Should we send a optin e-mail
- **data** (*struct*) – A struct of subscriber data, see *Subscriber data*

Returns True on success.

Return type boolean

Raises 1001: Invalid e-mail address

Raises 1003: The user is already subscribed

Raises 1004: Invalid data field

Raises 1011: New subscriptions blocked for your account

Raises 2001: Invalid mailing list

Raises 2004: Opt-in not configured (on one or more lists).

Raises 2005: Invalid operation on archived list

mailer.**delete_subscriber**(*email_address*)

Note: This method is an alias for: ubivox.cancel_all_subscriptions()

---

mailer.**get_subscription_consent_details**(*subscriber*, *list*)
Retrieve the consent details for any active subscription by a subscriber.

Parameters

- **subscriber** (*string*) – The e-mail address of a subscriber.
- **list** (*integer*) – A mailing list ID.

Returns Consent details.

Return type string

Raises 1001: Invalid e-mail address

Raises 1002: The user is not subscribed

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

mailer.**has_active_subscription**(*email_address*, *list*)
Check if a given e-mail address has an active subscription on a given list

Parameters

- **email_address** (*string*) – The address of the subscriber
- **list** (*integer*) – A mailing list ID.

Returns True if found, false if not.

Return type boolean

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

mailer.**list_subscriptions**(*list*, *offset*)
Retrieve the first 1000 subscribers from the offset specified, ordered by time added.

Parameters

- **list** (*integer*) – A mailing list ID.
- **offset** (*integer*) – (0-indexed) Offset.

Returns

A list structs of subscriptions:

- `subscriber`: Email address of the subscriber.
- `data`: A struct of subscriber data, see *Subscriber data*
- `state`: The subscription state. One of the following: `pending` (Pending end-user authorization), `active` (Active and receiving deliveries), `suspended` (Suspended due to bounces), `unsubscribed` (Unsubscribed by user), `removed` (Unsubscribed by client), `deleted` (Deleted)
- `added`: When was the subscription added
- `activated`: When was the subscription added
- `unsubscribed`: When was the subscription cancelled
- `suspended`: When was the subscription suspended
- `test_recipient`: A boolean indicating if the subscriber is a test recipient on this list

Return type array

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `5003`: Invalid offset

Note: Subscribers may appear more than once if they have a history of older subscriptions.

---

mailer.**remove_all_subscriptions**(*email_address*)
Remove all subscriptions for a given subscriber (administrative unsubscription).

Parameters **email_address** (*string*) – The e-mail address of the subscriber.

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

mailer.**remove_subscription**(*email_address*, *lists*)
Remove a subscription for the lists specified (administrative unsubscription).

Parameters

- **email_address** (*string*) – The address of the subscriber
- **lists** (*array*) – A list of mailinglist IDs

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

Raises 1002: The user is not subscribed

Raises 2001: Invalid mailing list

Note: If the user was not subscribed some of the lists, the NotSubscribed fault is raised.

---

mailer.**set_subscription_consent_details**(*subscriber*, *list*, *consent_details*)
Updates the consent details for any active subscription by a subscriber.

Parameters

- **subscriber** (*string*) – The e-mail address of a subscriber.
- **list** (*integer*) – A mailing list ID.
- **consent_details** (*string*) – The consent details

Returns True on success

Return type boolean

Raises 1001: Invalid e-mail address

Raises 1002: The user is not subscribed

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

mailer.**set_subscription_reference**(*subscriber*, *list*, *reference*)
Updates the reference field for a given subscription.

Parameters

- **subscriber** (*string*) – The e-mail address of a subscriber.
- **list** (*integer*) – A mailing list ID.
- **reference** (*string*) – The reference to be used.

Returns True on success.

Return type boolean

Raises 1001: Invalid e-mail address

Raises 1002: The user is not subscribed

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

mailer.**set_subscription_testrecipient**(*subscriber*, *list*, *test_recipient*)
Updates the subscription with test recipient status.

Parameters

- **subscriber** (*string*) – The email address of a subscriber.
- **list** (*integer*) – A mailing list ID.
- **test_recipient** (*boolean*) – A boolean indicating if the subscriber should receive tests.

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

Raises `1002`: The user is not subscribed

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

3.3 `subscriber` methods

`mailer.`**`forget_subscriber`**(*subscriber*, *data*)
Invoke the EU GDPR Right To Be Forgotten for a subscriber.

Parameters

- **`subscriber`** (*`string`*) – The e-mail address of the existing subscriber
- **`data`** (*`list`*) – A list of any data field keys potentially containing PII that needs to be cleared

Returns True on success

Return type boolean

Raises `1009`: Invalid subscriber

Note: The `data` parameter may also use a couple of special values: Use the string `all` to clear all data fields. Or `null` or an empty list to not clear any.

`mailer.`**`get_stopped_subscribers`**()
Retrieve a list of stopped subscribers.

Returns

A list of structs representing deliveries this subscriber has received:

- `email`: The e-mail address of the subscriber.
- `cause`: The cause of the entry on the stop list, see *Stoplist causes*.
- `stopped`: When was the stop list entry added.

Return type array

`mailer.`**`get_stopped_subscribers_since`**(*since*)
Retrieve a list of stopped subscribers.

Parameters **`since`** (*`string`*) – Return only stopped subscribers since (YYYY-MM-DD HH:MM:SS).

Returns

A list of structs representing deliveries this subscriber has received:

- `email`: The e-mail address of the subscriber.
- `cause`: The cause of the entry on the stop list, see *Stoplist causes*.
- `stopped`: When was the stop list entry added.

Return type array

Raises `5001`: Invalid date/time format

mailer.**get_subscriber**(*subscriber*)
Fetches information about a given subscriber.

Parameters **subscriber** (*string*) – The e-mail address of a subscriber.

Returns

A struct of:

- subscriptions: A list of structs:
  - list_title: Mailing list title
  - list_id: Mailing list ID
  - state: The subscription state. One of the following: pending (Pending end-user authorization), active (Active and receiving deliveries), suspended (Suspended due to bounces), unsubscribed (Unsubscribed by user), removed (Unsubscribed by client), deleted (Deleted)
  - added: When was the subscription added
  - activated: When was the subscription added
  - unsubscribed: When was the subscription cancelled
  - suspended: When was the subscription suspended
  - test_recipient: A boolean indicating if the subscriber is a test recipient on this list
- data: A struct of subscriber data, see *Subscriber data*

Return type struct

Raises 1001: Invalid e-mail address

mailer.**get_subscriber_delivery_stats**(*subscriber*)
Fetches delivery statistics for a given subscriber.

Parameters **subscriber** (*string*) – The e-mail address of a subscriber.

Returns

A list of structs representing deliveries this subscriber has received:

- list_id: The maillist ID
- list_title: The maillist title
- delivery_id: The delivery ID
- delivery_subject: The delivery ID
- delivery_send_time: The delivery send time
- read: Has the subscriber read this newsletter?
- clicked: Has the subscriber clicked this newsletter?

Return type array

Raises 1001: Invalid e-mail address

Note: Entries may be delayed for up to 24 hours after a delivery has been sent.

---

mailer.**list_all_subscribers**()
Retrieve all subscribers (up to 10000 only).

Returns A list e-mail-adresses.

Return type array

`mailer.`**`list_all_subscribers_with_data`**`()`
Retrieve all subscribers (up to 10000 only).

Returns

A list of structs of:

- `email`: E-mail address of the subscriber
- `data`: Data associated with the subscriber, see *Subscriber data*

Return type array

`mailer.`**`list_subscribers`**`(`*offset*`)`
Retrieve the first 50 subscribers from the offset specified.

Parameters **`offset`** (*`integer`*) – (0-indexed) Offset.

Returns A list e-mail-adresses.

Return type array

Raises `5003`: Invalid offset

`mailer.`**`list_subscribers_with_data`**`(`*offset*`)`
Retrieve the first 50 subscribers from the offset specified.

Parameters **`offset`** (*`integer`*) – (0-indexed) Offset.

Returns

A list of structs of:

- `email`: E-mail address of the subscriber
- `data`: Data associated with the subscriber, see *Subscriber data*

Return type array

Raises `5003`: Invalid offset

`mailer.`**`set_subscriber_data`**`(`*subscriber*, *data*`)`
Updates the subscribers data.

Parameters

- **`subscriber`** (*`string`*) – The email address of a subscriber.
- **`data`** (*`struct`*) – A struct of subscriber data, see *Subscriber data*

Returns True on success.

Return type boolean

Raises `1001`: Invalid e-mail address

Raises `1004`: Invalid data field

`mailer.`**`stop_subscriber`**`(`*subscriber*`)`
Add a subscriber to the stop list by administrative request.

Parameters **`subscriber`** (*`string`*) – The email address of a subscriber.

Returns True on success.

Return type boolean

Raises 1001: Invalid e-mail address

mailer.**update_subscriber_email**(*subscriber*, *new_email*, *responsible_user*, *consent*)
Update the e-mail address of an existing subscriber

Parameters

- **subscriber** (*string*) – The e-mail address of the existing subscriber
- **new_email** (*string*) – The new e-mail address
- **responsible_user** (*integer*) – The ID of the Ubivox user responsible for this change (for logging purposes).
- **consent** (*boolean*) – Has the subscriber given consent to this change (must be true)

Returns True on success

Return type boolean

Raises 1001: Invalid e-mail address

Raises 1003: The user is already subscribed

Raises 1009: Invalid subscriber

Raises 1010: Missing consent

Raises 5013: Invalid user

3.4 delivery methods

mailer.**copy_delivery**(*delivery*, *new_subject*, *new_maillist*)
Make a copy of a (sent or draft) delivery.

Parameters

- **delivery** (*integer*) – ID of the delivery.
- **new_subject** (*string*) – Subject of the delivery.
- **new_maillist** (*integer*) – ID of the maillist, which the new delivery will be placed on.

Returns The ID of the new delivery.

Return type integer

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

Raises 3001: Invalid delivery

mailer.**create_delivery**(*subject*, *html_body*, *text_body*, *list*)
Create a new delivery for the the specified list.

Parameters

- **subject** (*string*) – Subject for the delivery.
- **html_body** (*string*) – The HTML body of the delivery including logic for merging.
- **text_body** (*string*) – The text body of the delivery including logic for merging.
- **list** (*integer*) – The mailing list to be used. (integer ID)

Returns The new delivery ID on success.

Return type integer

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `3004`: Problem with delivery content

Raises `5014`: Invalid subject

`mailer.`**`delete_delivery`**(*delivery*)
Deletes a delivery and everything related to it (statistics, ...).

Parameters **`delivery`** (*`integer`*) – ID of the delivery.

Returns True on success.

Return type boolean

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

`mailer.`**`get_delivery`**(*delivery*)
Get information about a delivery.

Parameters **`delivery`** (*`integer`*) – ID of the delivery.

Returns

A struct of:

- `id`: Id of the delivery.
- `subject`: Subject of the delivery.
- `send_time`: The send time of the delivery (if available)
- `html_body`: The HTML body of the delivery including logic for merging.
- `text_body`: The text body of the delivery including logic for merging.
- `archive_html_body`: The HTML body of an archive version of the newsletter.
- `list`: The maillist to be used. (integer ID)
- `state`: Delivery state. One of: `edit` (Editing), `standby` (Standby), `active` (Processing), `queued` (Delivering), `done` (Delivered)

Return type struct

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

`mailer.`**`get_delivery_stats`**(*delivery*)
Get the simple statistics of the delivery.

Parameters **`delivery`** (*`integer`*) – ID of the delivery.

Returns

A struct of:

- `delivered_ok`: Mails delivered.
- `delivered_fail`: Mails bounced.

- `views`: Total views.
- `views_unique`: Unique views (readers).
- `view_rate`: Total view rate.
- `view_rate_unique`: Unique view rate (readers).
- `clicks`: Total clicks.
- `clicks_unique`: Unique clicks.
- `click_rate_delivered`: Click rate (of delivered).
- `click_rate_views`: Click rate (of views).
- `click_rate_delivered_unique`: Unique click rate (of delivered).
- `click_rate_views_unique`: Unique click rate (of views).
- `online_views`: Total online views.
- `online_views_unique`: Unique online views.
- `complaints`: Total complaints.
- `complaint_rate`: Total complaint rate.
- `unsubscriptions`: Total unsubscriptions.
- `unsubscription_rate`: Total unsubscription rate.

Return type struct

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Note: All rates are represented as the XML-RPC `double` type.

`mailer.`**`get_delivery_template_content`**`(`*`delivery`*`)`
Get the template configuration objects for a delivery.

Parameters **`delivery`** (*`integer`*) – ID of the delivery.

Returns JSON encoded string of the content object.

Return type string

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

`mailer.`**`list_all_deliveries`**`()`
Get list of all deliveries.

Returns

A list of structs of:

- `id`: ID if the delivery. (integer ID)
- `subject`: Subject of the delivery.
- `send_time`: The send time of the delivery (if available)
- `list`: The list ID. (integer ID)

- `state`: Delivery state. One of: `edit` (Editing), `standby` (Standby), `active` (Processing), `queued` (Delivering), `done` (Delivered)

Return type array

Raises `2005`: Invalid operation on archived list

`mailer.`**`list_deliveries`**(*state*, *offset*)
Get list of deliveries (the first 50 after `offset`).

Parameters

- **`state`** (*`string`*) – Delivery state filter. One of: `edit` (Editing), `standby` (Standby), `active` (Processing), `queued` (Delivering), `done` (Delivered)
- **`offset`** (*`integer`*) – (0-indexed) Offset.

Returns

A list of structs of:

- `id`: ID if the delivery. (integer ID)
- `subject`: Subject of the delivery.
- `send_time`: The send time of the delivery (if available)
- `list`: The list ID. (integer ID)
- `state`: Delivery state. One of: `edit` (Editing), `standby` (Standby), `active` (Processing), `queued` (Delivering), `done` (Delivered)

Return type array

Raises `2005`: Invalid operation on archived list

Raises `5003`: Invalid offset

`mailer.`**`send_delivery`**(*delivery*, *send_time*)
Queues the delivery for delivery.

Parameters

- **`delivery`** (*`integer`*) – ID of the delivery.
- **`send_time`** (*`string`*) – When should the delivery start (YYYY-MM-DD HH:MM:SS).

Returns True on success.

Return type boolean

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3002`: Delivery not in edit state

Raises `4008`: Sender is not verified

Raises `5001`: Invalid date/time format

`mailer.`**`set_delivery_in_archive`**(*delivery*, *in_archive*)
Configures the visibility of the delivery in the archive.

Parameters

- **`delivery`** (*`integer`*) – ID of the delivery.
- **`in_archive`** (*`boolean`*) – Should the delivery be made available in the archive

Returns True on success.

Return type boolean

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3006`: Delivery not sent

Important: If used, it must be called after `send_delivery`.

`mailer.`**`set_delivery_sender`**(*delivery*, *sender_email*, *sender_name*)

Sets sender details for a delivery.

Parameters

- **`delivery`** (*`integer`*) – ID of the delivery.
- **`sender_email`** (*`string`*) – E-mail address of the proposed sender
- **`sender_name`** (*`string`*) – Name of the proposed sender

Returns True on success.

Return type boolean

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3002`: Delivery not in edit state

Raises `4008`: Sender is not verified

Raises `5022`: Invalid sender or duplicates found

Important: If used, it must be called before `send_delivery`.

Important: The sender must exist in Ubivox.

`mailer.`**`set_delivery_sender_rewrite_data_field`**(*delivery*, *data*)

Sets sender rewrite data field for a delivery.

Parameters

- **`delivery`** (*`integer`*) – ID of the delivery.
- **`data`** (*`string`*) – Data key of the data field

Returns True on success.

Return type boolean

Raises `1005`: Invalid data field key

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3002`: Delivery not in edit state

Important: If used, it must be called before `send_delivery`.

`mailer.`**`set_delivery_subset_rules`**(*delivery*, *subset_rules*)
Sets the subset (segmentation) rules for a delivery.

Parameters

- **`delivery`** (*`integer`*) – ID of the delivery.
- **`subset_rules`** (*`string`*) – JSON string of subset rules.

Returns True on success.

Return type boolean

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3002`: Delivery not in edit state

Raises `5008`: Invalid rule set

Raises `9995`: Account type upgrade needed (contact sales)

Important: If used, it must be called before `send_delivery`.

Note: The JSON string containing the rules can be obtained from the Ubivox system by making a segmentation rule set and using the keyboard shortcut `ctrl+shift+x`.

`mailer.`**`suspend_delivery`**(*delivery*)
Suspends a delivery in the standby state and reverts to the edit state.

Parameters **`delivery`** (*`integer`*) – ID of the delivery.

Returns True on success.

Return type boolean

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3003`: Delivery not in standby state

`mailer.`**`test_delivery`**(*delivery*, *subscriber*)
Send a test delivery to a test recipient.

Parameters

- **`delivery`** (*`integer`*) – ID of the delivery.
- **`subscriber`** (*`string`*) – The e-mail address of a test subscriber.

Returns True on success.

Return type boolean

Raises `1002`: The user is not subscribed

Raises `1008`: The subscriber is not a test recipient

Raises 2005: Invalid operation on archived list

Raises 3001: Invalid delivery

Raises 3002: Delivery not in edit state

Note: The test subscriber must exist and be configured as a test recipient on the list of the delivery. See: *set_subscription_testrecipient()*

---

mailer.**update_delivery**(*delivery*, *subject*, *html_body*, *text_body*, *list*)

Updates a delivery.

Parameters

- **delivery** (*integer*) – ID of the delivery.
- **subject** (*string*) – Subject for the delivery.
- **html_body** (*string*) – The HTML body of the delivery including logic for merging.
- **text_body** (*string*) – The text body of the delivery including logic for merging.
- **list** (*integer*) – The mailing list to be used. (integer ID)

Returns The delivery ID on success.

Return type integer

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

Raises 3001: Invalid delivery

Raises 3002: Delivery not in edit state

Raises 3004: Problem with delivery content

Raises 5014: Invalid subject

mailer.**update_delivery_template_content**(*delivery*, *subject*, *template_content*, *list*)

Updates a delivery with a new template content.

Parameters

- **delivery** (*integer*) – ID of the delivery.
- **subject** (*string*) – Subject for the delivery.
- **template_content** (*string*) – The content object encoded as JSON.
- **list** (*integer*) – The mailing list to be used. (integer ID)

Returns The delivery ID on success.

Return type integer

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

Raises 3001: Invalid delivery

Raises 3002: Delivery not in edit state

Raises 3004: Problem with delivery content

Raises 3008: Invalid template content

Raises `5014`: Invalid subject

3.5 `splittest` methods

`mailer.`**`send_splittest`**(*list*, *title*, *delivery_a*, *delivery_b*, *send_time_a*, *send_time_b*, *snapshot_a*, *snapshot_b*, *send_time_final_a*, *send_time_final_b*, *test_size*, *criteria_operator*, *criteria_variable*, *criteria_threshold*, *subset_rules*)

Create a new split test using two existing deliveries..

Parameters

- **`list`** (*`integer`*) – The mailing list to be used. (integer ID)
- **`title`** (*`string`*) – Internal indentification title of the split test
- **`delivery_a`** (*`integer`*) – ID of the A delivery.
- **`delivery_b`** (*`integer`*) – ID of the B delivery.
- **`send_time_a`** (*`string`*) – When should the A delivery start (YYYY-MM-DD HH:MM:SS).
- **`send_time_b`** (*`string`*) – When should the B delivery start (YYYY-MM-DD HH:MM:SS).
- **`snapshot_a`** (*`string`*) – When should the A delivery be snapshotted (YYYY-MM-DD HH:MM:SS).
- **`snapshot_b`** (*`string`*) – When should the B delivery be snapshotted (YYYY-MM-DD HH:MM:SS).
- **`send_time_final_a`** (*`string`*) – When should the final delivery start, if A wins (YYYY-MM-DD HH:MM:SS).
- **`send_time_final_b`** (*`string`*) – When should the final delivery start, if B wins (YYYY-MM-DD HH:MM:SS).
- **`test_size`** (*`integer`*) – Percentage of the recipients to be used as splits (A+B).
- **`criteria_operator`** (*`string`*) – Operator for the win criteria.
- **`criteria_variable`** (*`string`*) – Variable for the win criteria.
- **`criteria_threshold`** (*`integer`*) – Threshold (in percent) needed for B to win over A.
- **`subset_rules`** (*`string`*) – JSON string of subset rules, leave as empty string for none.

Returns The new split test ID on success.

Return type integer

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `3001`: Invalid delivery

Raises `3002`: Delivery not in edit state

Raises `5001`: Invalid date/time format

Raises `5008`: Invalid rule set

Raises `5015`: Invalid percentage

Raises `5016`: Invalid criteria variable

Raises `5017`: Invalid criteria operator

Raises `5018`: Invalid schedule

Raises `9995`: Account type upgrade needed (contact sales)

Refer to these knowledge base articles for details on split testing in Ubivox:

- https://kb.ubivox.com/en/kb/how-to-split-test-in-ubivox/
- https://kb.ubivox.com/en/kb/split-testing-introduction-and-general-information/

Supported criteria operators:

- `most`
- `fewest`

Supported criteria variables:

- `views`: Views
- `views_unique`: Views (unique)
- `online_views`: Online views
- `online_views_unique`: Online views (unique)
- `clicks`: Clicks
- `clicks_unique`: Clicks (unique)
- `bounces`: Bounces
- `unsubscriptions`: Unsubscriptions
- `complaints`: Complaints
- `target_hits`: Target hits

Schedule constraints:

- The send time of *A* must be before the send time of *B*
- The send time of *A* must be before the snapshot time of *A*
- The send time of *B* must be before the snapshot time of *B*
- The snapshot time of *A* and *B* must be before both the final send time of *A* and *B*

3.6 `maillist` methods

`mailer.`**`copy_maillist`**(*maillist*, *new_title*)

Make a copy of a list with the same list settings, draft newsletters and course stages.

Parameters

- **`maillist`** (*`integer`*) – The ID of the mailing list.
- **`new_title`** (*`string`*) – The name of the new mailing list.

Returns ID of the new list

Return type integer

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

`mailer.`**`course_list_stages`**(*maillist*)
Get information about stages on a course for a given mailing list.

Parameters **`maillist`** (*integer*) – The ID of the mailing list.

Returns

An ordered array of structs of:

- `id`: ID of the stage. (integer ID)
- `subject`: Subject of the stage.
- `next_stage_interval`: The interval before the next stage (in hours)

Return type array

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

`mailer.`**`create_maillist`**(*title*, *language*, *delivery_sender_name*, *delivery_sender_email*, *primary_data_fields*)
Create a new maillist.

Parameters

- **`title`** (*string*) – The name of the maillist.
- **`language`** (*string*) – The language of the maillist.
- **`delivery_sender_name`** (*string*) – Name of the sender on deliveries.
- **`delivery_sender_email`** (*string*) – E-mail addresss of the sender on deliveries.
- **`primary_data_fields`** (*array*) – A list of data fields primary for this maillist.

Returns The new maillist ID on success.

Return type integer

Raises `1004`: Invalid data field

Raises `5006`: Invalid language

`mailer.`**`delete_maillist`**(*maillist*)
Deletes a maillist.

Parameters **`maillist`** (*integer*) – The ID of the maillist

Returns True on success.

Return type boolean

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

`mailer.`**`get_maillist`**(*maillist*)
Get a struct of list details.

Parameters **`maillist`** (*integer*) – The ID of the maillist

Returns

A struct with:

- `title`: List title

- `url`: URL for the list details in Ubivox
- `stats`: List statistics, a list of structs:
  - `date`: The date
  - `new_subscriptions`: New subscriptions (on this day)
  - `unsubscribed`: Unsubscribed (on this day)
  - `removed`: Removed (on this day)
  - `unsubscribed_removed`: Unsubscribed or removed (on this day)
  - `suspended`: Suspended (on this day)
  - `growth`: Net growth (on this day)
  - `active_total`: Total active subscriptions (as of this day)
  - `pending_total`: Total pending subscriptions (as of this day)
  - `unsubscribed_total`: Total unsubscribed subscriptions (as of this day)
  - `removed_total`: Total removed subscriptions (as of this day)
  - `unsubscribed_removed_total`: Total unsubscribed or removed subscriptions (as of this day)
  - `suspended_total`: Total suspended subscriptions (as of this day)

Return type struct

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Note: List statistics are cached for up to half an hour.

`mailer.`**`get_maillist_meta`**`(`*maillist*`)`

Get a struct of list meta data.

param integer maillist The ID of the maillist

returns A struct with:

- `signup_terms`: Text which is shown above the signup form
- `allow_public_signups`: Boolean showing whether public signups is allowed

‘ * `public_archive`: Boolean showing whether the list has a public archive

- `public_title`: The list’s public title
- `description`: The public list description
- `logo`: The path to the list logo

rtype struct

raises `2001`: Invalid mailing list

raises `2005`: Invalid operation on archived list

mailer.**get_maillists**()

Note: This method is an alias for: ubivox.list_maillists()

mailer.**list_maillists**()
Get a list of available mailing lists.

Returns

A list of structs:

- id: Maillist ID
- title: The title of the mailing list

Return type array

mailer.**maillist_archive**(*maillist*, *count*)
Find the most recently archived newsletters for a given list.

Parameters

- **maillist** (*integer*) – The ID of the mailing list.
- **count** (*integer*) – The amount of newsletters to return.

Returns

An ordered (by newsletter send time in descending order) array of structs of:

- id: ID of the newsletter. (integer ID)
- subject: Subject of the newsletter.
- archive_url: URL to the newsletter in the archive.
- archive_html: HTML version of the archived newsletter.
- send_time: When was the newsletter sent.

Return type array

Raises 2001: Invalid mailing list

Raises 2005: Invalid operation on archived list

mailer.**maillist_archive_meta**(*maillist*, *count*)
Find the most recently archived newsletters for a given list (without the HTML body).

Parameters

- **maillist** (*integer*) – The ID of the mailing list.
- **count** (*integer*) – The amount of newsletters to return.

Returns

An ordered (by newsletter send time in descending order) array of structs of:

- id: ID of the newsletter. (integer ID)
- subject: Subject of the newsletter.
- archive_url: URL to the newsletter in the archive.
- send_time: When was the newsletter sent.

Return type array

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

`mailer.`**`update_maillist`**(*maillist*, *title*, *language*, *delivery_sender_name*, *delivery_sender_email*, *primary_data_fields*)

Updates an existing maillist.

Parameters

- **`maillist`** (*`integer`*) – The ID of the maillist.
- **`title`** (*`string`*) – The name of the maillist.
- **`language`** (*`string`*) – The language of the maillist.
- **`delivery_sender_name`** (*`string`*) – Name of the sender on deliveries.
- **`delivery_sender_email`** (*`string`*) – E-mail addresss of the sender on deliveries.
- **`primary_data_fields`** (*`array`*) – A list of data fields primary for this maillist.

Returns True on success.

Return type boolean

Raises `1004`: Invalid data field

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `5006`: Invalid language

`mailer.`**`update_maillist_optin_optout_settings`**(*maillist*, *optin_subject*, *optin_body*, *optin_success_url*, *optin_failure_url*, *optout_success_url*)

Updates the opt-in/opt-out settings of an existing maillist.

Parameters

- **`maillist`** (*`integer`*) – The ID of the maillist.
- **`optin_subject`** (*`string`*) – Subject of the opt-in e-mail.
- **`optin_body`** (*`string`*) – Body of the opt-in e-mail (remember `%(link)s` for the opt-in link).
- **`optin_success_url`** (*`string`*) – Success URL for opt-in.
- **`optin_failure_url`** (*`string`*) – Failure URL for opt-in.
- **`optout_success_url`** (*`string`*) – Success URL for opt-out.

Returns True on success.

Return type boolean

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

3.7 `email` methods

`mailer.`**`send_email`**(*recipient_email*, *send_time*, *reference*, *sender_email*, *subject*, *html_body*, *text_body*)

Create a new e-mail and schedule it for delivery at `send_time`.

Parameters

- **`recipient_email`** (*`string`*) – The recipient e-mail address.
- **`send_time`** (*`string`*) – The send time.
- **`reference`** (*`string`*) – A preferably unique reference for this e-mail.
- **`sender_email`** (*`string`*) – The sender e-mail address.
- **`subject`** (*`string`*) – The subject of the e-mail.
- **`html_body`** (*`string`*) – The HTML body of the e-mail.
- **`text_body`** (*`string`*) – The text body of the e-mail.

Returns The reference of the e-mail on success.

Return type string

Raises `1001`: Invalid e-mail address

Raises `5001`: Invalid date/time format

Raises `5004`: Invalid reference (only a-z, A-Z, 0-9, - and _ allowed)

Raises `5021`: Invalid or empty body

Raises `9995`: Account type upgrade needed (contact sales)

Raises `9996`: Not approved for this call

Hint: Leave the reference as the empty string to have Ubivox create one for you.

`mailer.`**`send_email_from_newsletter`**(*recipient_email*, *data*, *send_time*, *reference*, *sender_email*, *sender_name*, *subject*, *delivery*)

Create a new e-mail and schedule it for delivery at `send_time`.

Parameters

- **`recipient_email`** (*`string`*) – The recipient e-mail address.
- **`data`** (*`struct`*) – Extra data for the mail merging
- **`send_time`** (*`string`*) – The send time.
- **`reference`** (*`string`*) – A preferably unique reference for this e-mail.
- **`sender_email`** (*`string`*) – The sender e-mail address.
- **`sender_name`** (*`string`*) – The sender name.
- **`subject`** (*`string`*) – The subject of the e-mail.
- **`delivery`** (*`integer`*) – The (integer) ID of a newsletter in Ubivox

Returns The reference of the e-mail on success.

Return type string

Raises `1001`: Invalid e-mail address

Raises 3001: Invalid delivery

Raises 3004: Problem with delivery content

Raises 3005: Mail merging error

Raises 5001: Invalid date/time format

Raises 5004: Invalid reference (only a-z, A-Z, 0-9, - and _ allowed)

Raises 9995: Account type upgrade needed (contact sales)

Raises 9996: Not approved for this call

Hint: Leave the reference as the empty string to have Ubivox create one for you.

If the recipient is known to Ubivox, we will use their data as a base and overwrite the final mail merging context with any data you supply in data.

If any of sender_email, sender_name or subject is left blank, we will use the defaults from the list associated with the newsletter.

mailer.**send_email_rfc2822**(*recipient_email*, *send_time*, *reference*, *message*)

Create a new e-mail and schedule it for delivery at send_time.

Parameters

- **recipient_email** (*string*) – The recipient e-mail address.
- **send_time** (*string*) – The send time.
- **reference** (*string*) – A preferably unique reference for this e-mail.
- **message** (*string*) – The raw (RFC2822) message.

Returns The reference of the e-mail on success.

Return type string

Raises 1001: Invalid e-mail address

Raises 5001: Invalid date/time format

Raises 5004: Invalid reference (only a-z, A-Z, 0-9, - and _ allowed)

Raises 5019: Invalid encoding

Raises 9995: Account type upgrade needed (contact sales)

Raises 9996: Not approved for this call

Hint: Leave the reference as the empty string to have Ubivox create one for you.

---

mailer.**send_email_with_names**(*recipient_email*, *recipient_name*, *send_time*, *reference*, *sender_email*, *sender_name*, *subject*, *html_body*, *text_body*)

Create a new e-mail and schedule it for delivery at send_time.

Parameters

- **recipient_email** (*string*) – The recipient e-mail address.
- **recipient_name** (*string*) – The recipient name.
- **send_time** (*string*) – The send time.

- **reference** (*string*) – A preferably unique reference for this e-mail.
- **sender_email** (*string*) – The sender e-mail address.
- **sender_name** (*string*) – The sender name.
- **subject** (*string*) – The subject of the e-mail.
- **html_body** (*string*) – The HTML body of the e-mail.
- **text_body** (*string*) – The text body of the e-mail.

Returns The reference of the e-mail on success.

Return type string

Raises `1001`: Invalid e-mail address

Raises `5001`: Invalid date/time format

Raises `5004`: Invalid reference (only a-z, A-Z, 0-9, - and _ allowed)

Raises `5021`: Invalid or empty body

Raises `9995`: Account type upgrade needed (contact sales)

Raises `9996`: Not approved for this call

Hint: Leave the reference as the empty string to have Ubivox create one for you.

3.8 `media` methods

`mailer.`**media_delete** (*filename*)

Deletes a media file on the server.

Parameters **filename** (*string*) – Relative filename of the media file (including directories).

Returns True on success.

Return type boolean

Raises `5007`: Invalid filename

`mailer.`**media_upload** (*filename*, *data*)

Writes a media file on the server. Will overwrite existing files.

Parameters

- **filename** (*string*) – Relative filename of the media file (including directories).
- **data** (*base64*) – The data of the media file.

Returns True on success.

Return type boolean

Raises `5007`: Invalid filename

3.9 data methods

mailer.**create_data_field**(*key*, *title*, *data_type*, *access*, *required*, *sort*, *archive_value*)
Create a data field.

Parameters

- **key** (*string*) – The key of the data field.
- **title** (*string*) – The title of the data field.
- **data_type** (*string*) – The data type. One of the following: `text` (Text field: Single line), `textarea` (Text field: Multi line), `checkbox` (Checkbox: Single), `select` (Select: Single), `select_multiple` (Select: Multiple), `select_radio` (Select: Radio buttons (single)), `select_checkbox` (Select: Check boxes (multiple))
- **access** (*string*) – Access level: `public` (Shown to users) or `private` (Only visible to administrators).
- **required** (*boolean*) – Is this field required.
- **sort** (*integer*) – An integer indicating the sort order.
- **archive_value** (*string*) – Default value for archive displays

Returns True on success.

Return type boolean

Raises `1005`: Invalid data field key

Raises `1006`: Invalid data field type

Raises `1007`: Invalid data field access

mailer.**get_data_keys**()

Note: This method is an alias for: `ubivox.list_data_fields()`

mailer.**hide_data_field**(*key*)
Hide the data field.

Parameters **key** (*string*) – The key of the data field

Returns True on success.

Return type boolean

Raises `1004`: Invalid data field

mailer.**list_data_fields**()
Retrieve the available data fields for user data.

Returns A list of keys.

Return type array

mailer.**list_data_fields_details**()
Retrieve the available data fields for user data with data field details.

Returns

A list of structs of:

- `id` - A unique integer ID for the data field
- `key` - The key of the data field
- `title` - The title of the data field
- `datatype` - The type of the data field (currenly only `text` (single line text field))
- `access` - Who can see the data field (`public` or `private`)
- `required` - Is the data field required (boolean)
- `archive_value` - The archive value of the data field
- `choices` - Predefined choices for the data field
- `sort` - Sort order (priority)

Return type struct

`mailer.`**`update_data_field`**(*key*, *title*, *data_type*, *access*, *required*, *sort*, *archive_value*)

Update a data field.

Parameters

- **`key`** (*`string`*) – The key of the data field (cannot be changed).
- **`title`** (*`string`*) – The new title of the data field.
- **`data_type`** (*`string`*) – The data type. One of the following: `text` (Text field: Single line), `textarea` (Text field: Multi line), `checkbox` (Checkbox: Single), `select` (Select: Single), `select_multiple` (Select: Multiple), `select_radio` (Select: Radio buttons (single)), `select_checkbox` (Select: Check boxes (multiple))
- **`access`** (*`string`*) – Access level: `public` (Shown to users) or `private` (Only visible to administrators).
- **`required`** (*`boolean`*) – Is this field required.
- **`sort`** (*`integer`*) – An integer indicating the sort order.
- **`archive_value`** (*`string`*) – Default value for archive displays

Returns True on success.

Return type boolean

Raises `1004`: Invalid data field

Raises `1006`: Invalid data field type

Raises `1007`: Invalid data field access

`mailer.`**`update_data_field_options`**(*key*, *options*)

Update a data field.

Parameters

- **`key`** (*`string`*) – The key of the data field to be updated
- **`options`** (*`array`*) – A list of structs with:
  - `order` - Sort order (integer)
  - `key` - The key of the option (string, what will be saved)
  - `value` - The value of t he option (string, what will be displayed)

Returns True on success.

Return type boolean

Raises `1004`: Invalid data field

Hint: Leave the *options* parameter as an empty list, to remove all options.

3.10 `imports` methods

`mailer.`**`import_history`**`()`

Retrieve a list of past imports. Refer to the web interface for details.

Returns

A list of structs:

- `filename` - Filename of the import
- `uploaded` - Date and time for the upload
- `imported` - Date and time when finished
- `maillist` - Maillist ID import to

Return type array

`mailer.`**`import_new`**`(`*filename*, *data*, *maillist*, *consent_details*, *update_data*, *ignore_empty_data*, *activate_suspended*, *unsubscribe_other*`)`

Create new import job and starts the processing immediately.

Parameters

- **`filename`** (*`string`*) – The filename of the import.
- **`data`** (*`base64`*) – The data of the import file.
- **`maillist`** (*`integer`*) – The ID of the maillist the subscriptions should be imported to.
- **`consent_details`** (*`string`*) – The cirsumstances under which consent was given for these addresses. Will be public to the subscriber.
- **`update_data`** (*`boolean`*) – Update data on existing subscriptions?
- **`ignore_empty_data`** (*`boolean`*) – Ignore empty values when updating data fields?
- **`activate_suspended`** (*`boolean`*) – Activate suspended subscriptions?
- **`unsubscribe_other`** (*`boolean`*) – Unsubscribe subscribers not in the import file?

Returns True on success.

Return type boolean

Raises `1005`: Invalid data field key

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `4002`: Too many invalid addresses in import file

Raises `4003`: Another import already exists with this filename

Raises `4004`: Invalid import filename

> **Warning:** A common pitfall of this method is that the `data` parameter must be of the `base64` XML-RPC data type, and not just a base64 encoded string. Refer to your XML-RPC library documentation for implementation notes.

`mailer.`**`import_queue`**`()`
Retrieve a list of currently processing and queued imports. Refer to the web interface for details.

Returns

A list of structs:

- `filename` - Filename of the import
- `uploaded` - Date and time for the upload
- `started` - Date and time when processing started (0000-00-00T00:00:00 if not started yet)
- `maillist` - Maillist ID import to
- `progress` - The progress expressed as an integer percentage

Return type array

3.11 `exports` methods

`mailer.`**`export_download`**`(`*`export`*`)`
Downloads a finished export.

Parameters **`export`** (*`integer`*) – The ID of the export to download.

Returns

A struct containing:

- `filename` - The filename of the export
- `format` - The format of the export (`xls` or `csv`)
- `contents` - The contents of the export

Return type struct

Raises `4006`: Invalid export

Raises `4007`: Export is not done processing yet

Note: The contents of the file is sent as a `base64` XML-RPC data type. Refer to your XML-RPC library documentation for implementation notes.

`mailer.`**`export_history`**`()`
Retrieve a list of past exports. Refer to the web interface for details.

Returns

A list of structs:

- `id` - ID of the export
- `maillist` - Maillist ID
- `title` - Title of the export

- `start_time` - When was the export started
- `format` - `csv` or `xls`
- `state` - `ordered`, `processing` or `done`

Return type array

`mailer.`**`export_order`**(*maillist*, *title*, *start_time*, *format*, *data*, *export_active*, *export_suspended*, *export_unsubscribed*, *export_removed*)

Order a new export. Refer to the web interface for details.

Parameters

- **`maillist`** (*integer*) – The ID of the maillist containing subscriptions for this export.
- **`title`** (*string*) – The title of the export (for identification).
- **`start_time`** (*string*) – When can we start processing the export (YYYY-MM-DD HH:MM:SS).
- **`format`** (*string*) – Either `csv` or `xls`.
- **`data`** (*array*) – An array of data keys to include in the export.
- **`export_active`** (*boolean*) – Include active subscriptions in export.
- **`export_suspended`** (*boolean*) – Include suspended subscriptions in export.
- **`export_unsubscribed`** (*boolean*) – Include unsubscribed subscription in export.
- **`export_removed`** (*boolean*) – Include removed subscriptions in export.

Returns New export ID on success

Return type integer

Raises `1004`: Invalid data field

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `5001`: Invalid date/time format

3.12 `target` methods

`mailer.`**`create_target`**(*title*, *description*)

Create a new target.

Parameters

- **`title`** (*string*) – The name of the target.
- **`description`** (*string*) – The description of the target.

Returns The new target ID on success.

Return type integer

`mailer.`**`list_targets`**()

Get a list of available targets.

Returns

A list of structs:

- `id`: Target ID
- `title`: The title of the target
- `description`: The description of the target

Return type array

`mailer.`**`target_statistics`**(*target*)
Fetches statistics for an existing target.

Parameters **`target`** (*integer*) – The ID of the target.

Returns

A struct of:

- `title`: Title of the target.
- `description`: Description of the target.
- `hits`: Hits to this target.
- `time`: Average time in seconds to reach target.

Return type struct

Raises `5010`: Invalid target

`mailer.`**`update_target`**(*target*, *title*, *description*)
Updates an existing target.

Parameters

- **`target`** (*integer*) – The ID of the target.
- **`title`** (*string*) – The name of the target.
- **`description`** (*string*) – The description of the target.

Returns True on success.

Return type bool

Raises `5010`: Invalid target

3.13 `webhook` methods

`mailer.`**`create_webhook`**(*hook*, *url*, *throttle_per_minute*, *max_retries*)
Create a new webhook.

Parameters

- **`hook`** (*string*) – The hook to attach to, see *Supported hooks for webhooks*.
- **`url`** (*string*) – The URL to call.
- **`throttle_per_minute`** (*integer*) – How many calls will we attempt per minute.
- **`max_retries`** (*integer*) – How many times will we retry a call.

Returns The (integer) ID of the new webhook on success.

Return type int

Raises `5012`: Invalid hook

mailer.**delete_webhook**(*id*)
Delete an existing webhook.

Parameters **id** (*integer*) – The ID of the webhook.

Returns True on success.

Return type boolean

Raises 5011: Invalid webhook

mailer.**delete_webhook_by_url**(*url*)
Delete an existing webhook.

Parameters **url** (*string*) – The URL of the webhook(s).

Returns True on success.

Return type boolean

Raises 5011: Invalid webhook

mailer.**list_webhooks**()
Retrieve the available, configured webhooks.

Returns

A list of structs of:

- id - A unique integer ID for the webhook
- hook - The configured hook
- url - The configured URL to call
- throttle_per_minute - How many calls will we attempt per minute
- max_retries - How many times will we retry a call

Return type array

mailer.**update_webhook**(*id*, *hook*, *url*, *throttle_per_minute*, *max_retries*)
Update an existing webhook.

Parameters

- **id** (*integer*) – The ID of the webhook.
- **hook** (*string*) – The hook to attach to, *Supported hooks for webhooks*.
- **url** (*string*) – The URL to call.
- **throttle_per_minute** (*integer*) – How many calls will we attempt per minute.
- **max_retries** (*integer*) – How many times will we retry a call.

Returns True on success.

Return type boolean

Raises 5011: Invalid webhook

Raises 5012: Invalid hook

3.14 `account` methods

`mailer.`**`account_flag`**(*flag*)
Check the status of a given account flag.

Parameters **`flag`** (*string*) – The name of the flag. See below.

Returns Contents of the flag or the empty string for unknown flags.

Return type string

Currently supported flags:

- `has_verified_sender`

  `yes` or `no` depending on whether the account has Verified Sender enabled.

- `verified_sender_cert_url`

  The URL for the verified sender certificate or the empty string if Verified Sender is not enabled.

`mailer.`**`account_invoice_details`**(*invoice_number*)
List all invoices on account

Returns

Struct with:

- `number`: Number (string)
- `date`: Date (string)
- `currency`: Currency (string, ISO 4217)
- `paid`: Paid (boolean)
- `paid_text`: Text describing the payment (string)
- `paid_amount`: How much was paid (double)
- `paid_fee`: How much was paid in fees (double)
- `paid_date`: On what day was the payment fulfilled (string)
- `invoice_lines`: Array invoice lines as structs with:
  - `product_number`: Product number (string)
  - `quantity`: Quantity (integer)
  - `unit`: Unit (string)
  - `unit_net_amount`: Net amount per unit (double)
  - `total_net_amount`: Total net amount for line (double)
  - `taxes`: Any value added taxes related to this line as array of structs with:
    - `name`: Name of the tax (string)
    - `percentage`: Tax percentage (double)
    - `amount`: Amount to be paid in tax (double)
- `payments`: Array of payment receipts as structs with:
  - `opened`: Payment opened (string)
  - `confirmed`: Payment confirmed (string)

- `fee`: The payment fee for this payment (double)
- `amount`: Amount paid (double)
- `reference`: Reference (string)

- `net_amount`: Net amount (string)
- `vat_amount`: VATs amount (string)
- `gross_amount`: Gross amount (string)

Return type struct

Raises `5020`: Invalid invoice

`mailer.`**`account_invoice_list`**`()`
List all invoices on account

Returns

Array of structs with:

- `number`: Number (string)
- `date`: Date (string)
- `due_date`: Payment due date (string)
- `currency`: Currency (string, ISO 4217)
- `gross_amount`: Gross amount (string)
- `paid`: Paid (boolean)

Return type array

`mailer.`**`account_invoice_pdf`**`(`*`invoice_number`*`)`
Fetch an invoice as PDF

Parameters **`invoice_number`** (*`integer`*) – Invoice number

Returns The invoice as PDF (as XML-RPC base64 encoded binary).

Return type base64

Raises `5020`: Invalid invoice

`mailer.`**`account_status`**`()`
Get an account status struct suitable for producing an account dashboard.

Returns

struct:

- `lists`: Your lists (limited to 25)
  - `title`: List title
  - `url`: URL for the list details in Ubivox
  - `stats`: List statistics, a list of structs:
    - `date`: The date
    - `new_subscriptions`: New subscriptions (on this day)
    - `unsubscribed`: Unsubscribed (on this day)
    - `removed`: Removed (on this day)

        * `unsubscribed_removed`: Unsubscribed or removed (on this day)
        * `suspended`: Suspended (on this day)
        * `growth`: Net growth (on this day)
        * `active_total`: Total active subscriptions (as of this day)
        * `pending_total`: Total pending subscriptions (as of this day)
        * `unsubscribed_total`: Total unsubscribed subscriptions (as of this day)
        * `removed_total`: Total removed subscriptions (as of this day)
        * `unsubscribed_removed_total`: Total unsubscribed or removed subscriptions (as of this day)
        * `suspended_total`: Total suspended subscriptions (as of this day)
    - `drafts`: Your newsletter drafts (latest 25 only)
      - `subject`: Subject of the newsletter
      - `edit_time`: Last edited time
      - `url`: URL for the newsletter details in Ubivox
      - `list_title`: The title of the list of the newsletter
      - `list_url`: URL the the list details of the list of the newsletter
    - `published`: Your published, sent newsletters (latest 25 only)
      - `subject`: Subject of the newsletter
      - `send_time`: Last edited time
      - `url`: URL for the newsletter details in Ubivox
      - `list_title`: The title of the list of the newsletter
      - `list_url`: URL the the list details of the list of the newsletter
      - `recipients`: Total number of recipients
      - `delivered`: Total number of successful deliveries
      - `views`: Total number of views (opens) on this newsletter
      - `clicks`: Total number of clicks on this newsletter

  Return type struct

Note: List statistics are cached for up to half an hour.

3.15 `sync` methods

`mailer.`**`synchronize`**(*new_subscriptions*, *maillist*, *optin*, *unsubscriptions_from*)

Working with your local CRM or database system as master, synchronize new subscriptions to Ubivox while getting information about unsubscriptions from Ubivox as well.

Parameters

- **`new_subscriptions`** (*array*) – An array (max length 500) of structs with:

  - `email` - The e-mail address of the subscriber.
  - `time` - The time of the new subscription (optional, see note)
  - `data` - A struct of data fields to be updated. See *Subscriber data*.
  - `consent_details` - A text describing the circumstances under which consent was given.
- **`maillist`** (*`integer`*) – The list IDs for the subscribers to be subscribed to. Also has effect on the returned unsubscriptions.
- **`optin`** (*`boolean`*) – Should we send opt-in e-mails for this synchronization call.
- **`unsubscriptions_from`** (*`string`*) – Return unsubscriptions from this point in time. (Leave as empty string to disable)

Returns

A struct with:

- `status` - An array of structs with details regarding the new subscriptions:
  - `email` - The e-mail address of the subscriber.
  - `subscriber_id` - The subscriber id, may be NULL if the e-mail address is considered invalid.
  - `subscription_id` - The resulting subscription id, may be NULL if we ignored the request due to block lists or existing unsubscriptions.
  - `message` - A human readable message detailing what we did with this subscriber.
- `unsubscriptions` - An array of structs with details regarding unsubscriptions that occurred since the specified time:
  - `email` - The e-mail address of the subscriber.
  - `subscriber_id` - The subscriber id.
  - `subscription_id` - The ID of the unsubscription.
  - `time` - Time stamp for the unsubscription.

Return type struct

Raises `1011`: New subscriptions blocked for your account

Raises `2001`: Invalid mailing list

Raises `2005`: Invalid operation on archived list

Raises `5001`: Invalid date/time format

Note: If you do not include a subscription time in the `new_subscriptions` structs, the presence of an earlier unsubscription will cause us to ignore the new subscription request. If you include it, and the new subscription time is later than the unsubscription time we have on file, the new subscription will be allowed.

This method is meant to be run using cron or a similar mechanism, synchronizing your new subscription from your system to Ubivox and also fetching unsubscriptions from Ubivox to your system. We recommend running it in daily or hourly schedules, depending on your load and needs.

If you need one-time batch imports of large subscriber sets, we recommend you use the import API, see *imports methods*.

This method is *idempotent* in the sense that the system will ignore requests to subscribe already subscribed adresses and the returned unsubscriptions doesn't have any side effects on the system.

Status messages

The returned status messages can be:

- `Invalid e-mail address.`

  The e-mail address didn't validate or was present on a blacklist.

- `Subscriber on stop list.`

  The subscriber is on your stop list and cannot be added without using opt-in.

- `Suspended subscriber.`

  The subscriber is suspended on one or more of your lists.

- `Already subscribed (state).`

  The subscriber is already subscribed to your list in the state `state`.

- `Subscription created.`

  A new subscription was successfully created for this subscriber.

Warning: It is the responsibility of the caller to archive the status messages in order to debug why a given subscriber was blocked.

3.16 `ecommerce` methods

`mailer.`**`ecommerce_tracking_cancel_order`**(*customer_ref*, *order_ref*)
Cancels an order previously registered.

Parameters

- **`customer_ref`** (*`string`*) – Your customer reference
- **`order_ref`** (*`string`*) – Your order reference

Returns True on success, False if the order wasn't found

Return type boolean

Raises `9995`: Account type upgrade needed (contact sales)

Note: Refer to the *Ecommerce Tracking* documentation for details.

`mailer.`**`ecommerce_tracking_html`**(*customer_ref*, *order_ref*)
Register new sale for a contact.

Parameters

- **`customer_ref`** (*`string`*) – Your customer reference
- **`order_ref`** (*`string`*) – Your order reference

Returns A HTML snippet for pinging your ecommerce tracking data from the client side potentially allowing to associate the sale with a click/newsletter

Return type string

Raises `9995`: Account type upgrade needed (contact sales)

Note: Refer to the *Ecommerce Tracking* documentation for details.

`mailer.`**`ecommerce_tracking_new_order`**(*params*, *email_address*, *maillist*)
Register new sale for a contact.

Parameters

- **`params`** (*`struct`*) – Ecommerce tracking order parameters.
- **`email_address`** (*`string`*) – E-mail address of the contact
- **`maillist`** (*`integer`*) – To associate the sale with a specific list, pass the maillist ID here. Leave as 0 for no maillist association.

Returns "OK" for success. In case of problems parsing your ecommerce tracking parameters, an error message will be returned.

Return type string

Raises `2001`: Invalid mailing list

Raises `9995`: Account type upgrade needed (contact sales)

Note: Refer to the *Ecommerce Tracking* documentation for details on the order parameters.

3.17 `misc` methods

`mailer.`**`ping`**`()`

Note: This method is an alias for: `ubivox.server_time()`

`mailer.`**`server_time`**`()`
Returns the current server time.

Returns The server time

Return type string

`mailer.`**`time_zone`**`()`
Get information on the current time zone of the account.

Returns The time zone name.

Return type string

`mailer.`**`time_zones`**`()`
Get information on the supported time zones.

Returns An array of supported time zones

Return type array

CHAPTER

FOUR

PARTNER API METHODS

These methods are only supported for partner accounts. Read more about Ubivox partner accounts on the website.

Note: Refer to your partner manual for the specific semantics regarding client states and approval process. The partner manual can be downloaded from within the Ubivox system by selecting *Partner* then *Partner tools*.

4.1 `partner` methods

`partner.`**`activate_client`**`(`*client*`)`
Activate operation of a client.

Parameters **`client`** (*`integer`*) – ID of the client.

Returns true

Return type boolean

Raises `7001`: Invalid client

Raises `7009`: Missing address field for customer

`partner.`**`approve_client`**`(`*client*`)`
Approve a client.

Parameters **`client`** (*`integer`*) – ID of the client.

Returns true

Return type boolean

Raises `7001`: Invalid client

Raises `7009`: Missing address field for customer

`partner.`**`available_account_types`**`()`
Retrieves an array of the available account types with IDs suitable for other partner calls.

Returns

An array of structs of:

- `id` - ID of the account type
- `name` - Name of the account type

Return type struct

partner.**client_details**(*client*)

Fetches contact information for an existing client in Ubivox.

Parameters **client** (*integer*) – The ID of the existing client.

Returns

A struct of:

- notification_language - Primary language for the client (da, en or es).
- time_zone - Primary time zone for the client (e.g.: Europe/Copenhagen).
- contact_name - Name of the primary contact person for the client.
- contact_email - E-mail address of the primary contact person for the client.
- contact_phone - Phone number of the primary contact person for the client.
- company_address_company_name - (Company address) Company name.
- company_address_vat_number - (Company address) VAT number.
- company_address - (Company address) Address (multiple lines).
- company_address_zip - (Company address) Zip code.
- company_address_city - (Company address) City.
- company_address_phone - (Company address) Phone number.
- company_address_country - (Company address) Country (ISO 3166-1 alpha-2 upper case).
- billing_address_billing_name - (Billing address) Company name.
- billing_address_vat_number - (Billing address) VAT number.
- billing_address - (Billing address) Address (multiple lines).
- billing_address_zip - (Billing address) Zip code.
- billing_address_city - (Billing address) City.
- billing_address_phone - (Billing address) Phone number.
- billing_address_country - (Billing address) Country (ISO 3166-1 alpha-2 upper case).

Return type struct

Raises 7001: Invalid client

partner.**create_client**(*account_name*, *language*, *time_zone*, *account_type*, *contact_name*, *contact_email*, *contact_phone*, *company_address_company_name*, *company_address_vat_number*, *company_address*, *company_address_zip*, *company_address_city*, *company_address_phone*, *company_address_country*, *billing_address_company_name*, *billing_address_vat_number*, *billing_address*, *billing_address_zip*, *billing_address_city*, *billing_address_phone*, *billing_address_country*)

Create a new client in Ubivox.

The client will be created in a *suspended* and *unapproved* state.

Parameters

- **account_name** (*string*) – The account name (will be used for hostnames, usernames).
- **language** (*string*) – Primary language for the new client (da, en or es).

- **time_zone** (*string*) – Primary time zone for the new client (e.g.: Europe/Copenhagen).
- **account_type** (*integer*) – The ID of the account type to be used for the new client.
- **contact_name** (*string*) – Name of the primary contact person for the new client.
- **contact_email** (*string*) – E-mail address of the primary contact person for the new client.
- **contact_phone** (*string*) – Phone number of the primary contact person for the new client.
- **company_address_company_name** (*string*) – (Company address) Company name.
- **company_address_vat_number** (*string*) – (Company address) VAT number.
- **company_address** (*string*) – (Company address) Address (multiple lines).
- **company_address_zip** (*string*) – (Company address) Zip code.
- **company_address_city** (*string*) – (Company address) City.
- **company_address_phone** (*string*) – (Company address) Phone number.
- **company_address_country** (*string*) – (Company address) Country (ISO 3166-1 alpha-2 upper case).
- **billing_address_billing_name** (*string*) – (Billing address) Company name.
- **billing_address_vat_number** (*string*) – (Billing address) VAT number.
- **billing_address** (*string*) – (Billing address) Address (multiple lines).
- **billing_address_zip** (*string*) – (Billing address) Zip code.
- **billing_address_city** (*string*) – (Billing address) City.
- **billing_address_phone** (*string*) – (Billing address) Phone number.
- **billing_address_country** (*string*) – (Billing address) Country (ISO 3166-1 alpha-2 upper case).

Returns

A struct with:

- `id`: The ID of the new client.
- `base_url`: Base URL of the new client.
- `username`: Username of the first, auto generated, user, for the API and feeds.
- `password`: Password of the first, auto generated, user
- `api_password`: An auto generated API password for API access to this client.
- `feeds_password`: An auto generated feeds password for feeds access to this client.

Return type struct

Raises `5006`: Invalid language

Raises `5009`: Invalid timezone

Raises `7006`: Account name taken

Raises `7007`: Invalid account type

Raises `7008`: Invalid account name

Raises `7009`: Missing address field for customer

`partner.`**`disapprove_client`**(*client*)
Disapprove a client.

Parameters **`client`** (*integer*) – ID of the client.

Returns true

Return type boolean

Raises `7001`: Invalid client

`partner.`**`list_clients`**()
Fetch a list of your clients on the Ubivox system

Returns

array of struct of:

- `id`: The ID of the client.
- `base_url`: Base URL of the client.
- `company_address_company_name`: Company address: Company name
- `billing_address_company_name`: Billing address: Company name
- `contact_name`: Primary contact name
- `contact_email`: Primary contact e-mail
- `contact_phone`: Primary contact phone
- `state`: Client state (`active` or `suspended`)
- `approved`: Approval status (boolean)

Return type array

`partner.`**`sudo`**(*client*, *method*, *params*)
Execute a method as one of your clients without configuring a new API password for each account.

Parameters

- **`client`** (*integer*) – The ID of the client to run the method as.
- **`method`** (*string*) – The full method name (e.g.: `ubivox.ping`).
- **`params`** (*array*) – The parameters for the method (use an empty array for no parameters).

Returns The return value of the method used.

Return type mixed

Raises `7001`: Invalid client

Raises `7010`: Invalid method

Raises May raise any error from the called method.

Note: Supports only methods in the `ubivox.` and `system.` namespace.

`partner.`**`suspend_client`**(*client*)
Suspend operation of a client.

Parameters **`client`** (*integer*) – ID of the client.

Returns true

Return type boolean

Raises `7001`: Invalid client

`partner.`**`update_client`**(*client*, *language*, *time_zone*, *contact_name*, *contact_email*, *contact_phone*, *company_address_company_name*, *company_address_vat_number*, *company_address*, *company_address_zip*, *company_address_city*, *company_address_phone*, *company_address_country*, *billing_address_company_name*, *billing_address_vat_number*, *billing_address*, *billing_address_zip*, *billing_address_city*, *billing_address_phone*, *billing_address_country*)

Updates contact information for an existing client in Ubivox.

Parameters

- **`client`** (*`integer`*) – The ID of the existing client.
- **`language`** (*`string`*) – Primary language for the new client (da, en or es).
- **`time_zone`** (*`string`*) – Primary time zone for the new client (e.g.: Europe/Copenhagen).
- **`contact_name`** (*`string`*) – Name of the primary contact person for the new client.
- **`contact_email`** (*`string`*) – E-mail address of the primary contact person for the new client.
- **`contact_phone`** (*`string`*) – Phone number of the primary contact person for the new client.
- **`company_address_company_name`** (*`string`*) – (Company address) Company name.
- **`company_address_vat_number`** (*`string`*) – (Company address) VAT number.
- **`company_address`** (*`string`*) – (Company address) Address (multiple lines).
- **`company_address_zip`** (*`string`*) – (Company address) Zip code.
- **`company_address_city`** (*`string`*) – (Company address) City.
- **`company_address_phone`** (*`string`*) – (Company address) Phone number.
- **`company_address_country`** (*`string`*) – (Company address) Country (ISO 3166-1 alpha-2 upper case).
- **`billing_address_billing_name`** (*`string`*) – (Billing address) Company name.
- **`billing_address_vat_number`** (*`string`*) – (Billing address) VAT number.
- **`billing_address`** (*`string`*) – (Billing address) Address (multiple lines).
- **`billing_address_zip`** (*`string`*) – (Billing address) Zip code.
- **`billing_address_city`** (*`string`*) – (Billing address) City.
- **`billing_address_phone`** (*`string`*) – (Billing address) Phone number.
- **`billing_address_country`** (*`string`*) – (Billing address) Country (ISO 3166-1 alpha-2 upper case).

Returns true

Return type boolean

Raises `5006`: Invalid language

Raises `5009`: Invalid timezone

Raises `7001`: Invalid client

Raises `7009`: Missing address field for customer

CHAPTER

FIVE

SALES TRACKING

5.1 Background

The Sales tracking API works by embedding different tracking 1x1 pixel images (Targets) on your website at different strategic places.

Points where these targets would make sense is often at the end of a given transaction. That could be:

- After a completed signup (e.g. to an event or competition)
- After a completed sale

The targets can be configured through the Ubivox system by going to *Statistics* then *Targets* in the top menu.

5.2 Adding sales numbers

If you are tracking sales, you might want to track the amounts in order to track the total sales or turnover generated by your newsletters.

Sales numbers can be added to the newsletters statistics by extending the normal image URL with these query string (GET) parameters:

Parameter	Description	Example
`st_revenue`	The revenue from the transaction	`st_revenue=49.95`
`st_quantity`	Items in the transaction	`st_quantity=2`
`st_currency`	Currency of the transaction	`st_currency=USD`
`st_reference`	Transaction reference (unique)	`st_reference=12345678`

5.2.1 Parameter details

- At least one of `st_revenue` or `st_quantity` is required
- Use period as decimal separator and do not use thousands separators. Decimals are optional. Valid examples: *5*, *49.95*, *14999.95*.
- Maximum trackable revenue amount is: *99999999.99*.
- The `st_reference` parameter is built to identify specific transactions or registrations hitting the target. The reference has to be unique (e.g. an order number); if we register multiple target hits with the same reference, only the first hit will be registered. The reference may be both numerical or text but has to be unique. (Maximum length: 100 characters)

5.2.2 Example

A full example of a target with added sales tracking details, could look like this:

```
<img src="https://example.clients.ubivox.com/target/<target_id>/?st_revenue=49.95&st_
↪quantity=2&st_currency=USD&st_reference=12345678" height="1" width="1">
```

5.3 Ecommerce Tracking

If your account supports the Ecommerce Tracking API, you gain the option of registering each individual product in the set of products for each order as well as being able to segment on those products.

5.3.1 Parameter details

There's multiple ways for the order data to enter the system, but all of them uses this common set of parameters.

Order parameters

et_products ***(required)*** How many products are in the order.

et_currency ***(required)*** What currency was the order made in. Use Upper case ISO 4217 three letter codes. Example: `USD`, `DKK` or `GBP`

et_customer_ref ***(required)*** Your customer reference. Should uniqely identify the customer at your end.

et_order_ref ***(required)*** Your order reference. Should uniqely identify the order at your end.

et_point_of_sale ***(optional)*** Annotate the order with the point of sale for the transaction. Example: `webshop`.

et_added ***(optional)*** If the order time is not simultaneous with the parameter generation, annotate the order with a RFC3339/ISO8601 timestamp to use for order If not time zone offset is included, UTC is assumed. Example: `2017-05-23T13:38:12+00:00`.

Product parameters

For each product in the order, you need these parameters. Replace `i` with a zero indexed counter.

et_prod_i_id ***(required)*** A unique product ID should never change for a product. If we see a different product (name, category) at a later time with the same product ID, we will update the earlier orders as well. This will allow you to propagate changes to the system.

et_prod_i_quantity ***(required)*** How many instances of the product was sold.

et_prod_i_name ***(required)*** The product name.

et_prod_i_category ***(required)*** A comma separated list of categories for this product.

et_prod_i_unit_price ***(required)*** The unit price of the product formatted as `12345.67`.

Example parameter set

Here is an example parameter set of an order with two products with a total of seven items.

Key	Value
et_customer_ref	HS99332
et_order_ref	995AB220
et_point_of_sale	webshop
et_currency	USD
et_products	2
et_prod_0_id	9040493329
et_prod_0_quantity	5
et_prod_0_category	socks,mens
et_prod_0_name	Mens black socks
et_prod_0_unit_price	0.89
et_prod_1_id	8584922943
et_prod_1_quantity	2
et_prod_1_category	tshirts,mens
et_prod_1_name	Mens red tshirt
et_prod_1_unit_price	1.19

5.3.2 Sending order data using the XML-RPC API

The first, and recommended option, is to send the order data using the XML-RPC API. A complete integration consists of the following steps:

1. On the order confirmation page (after the sale is done), call the *mailer.ecommerce_tracking_html()* API method to retrieve a HTML snippet that must be printed on the page.

   This call must be made for each order (may even be called multiple time per order) as this snippet will be configured for this particular customer and order. It will be both encrypted and tampering-proof, so you wont accidentally leak any details.

   The HTML snippet will be a small 1x1 transparent GIF image to allow us to read a tracking cookie that may have been set when your subscriber clicked a link in a newsletter on your account.

2. At your decision, call the API method *mailer.ecommerce_tracking_new_order()* either during the ordering process or at a later time. The sooner you call this, the sooner you will have the data available in the system for segmentation and analytics.

   The `params` parameter of the XML-RPC method, is an XML-RPC struct with the above parameters.

   This step is completely server side and thus is not prone to blocking or tampering by the client.

3. In case your customer cancels their order, consider marking it as such in our data as well, so you don't risk ending up sending out unsolicited follow-up e-mails for your customer's orders.

   This is done by calling *mailer.ecommerce_tracking_cancel_order()* as soon as possible after the order has been cancelled.

If you complete thise steps, we will be able to associate the sale with the:

- Contact
- Subscription
- List

- Newsletter
- Link

The contact does not need to have a subscription to your newsletter yet. We will still store the data in your account and if the contact at some point subscribes to your newsletter, you will have all of the sales history available.

It is possible to only implement step 2, but then you will not be able to track sales to specific newsletters and links. Furthermore, if you don't implement step 3, you might risk accidentally sending unwanted follow-up e-mails to contacts that cancelled their orders.

Note: As with all calls to remote systems, please consider what happens if the API is unavailable and make sure your end handles the failure gracefully. Your customer should be able to complete their order even in the rare circumstance that the API is unavailble. See *Service windows*

5.3.3 Sending order data using tracking image

The second option is to use only the tracking image to transfer all parameters of the order to the system in an URL encoded query string.

In this mode, we will have the contact details for your subscriber available in a tracking cookie sent when they last clicked on a link in your newsletter, and we will associate the sale with the:

- Contact
- Subscription
- List
- Newsletter
- Link

This also means that, if a sale happens and the user has not clicked on a link in your newsletter or is blocking our cookies, the sale will not be recorded.

The tracking GIF is available on your account from:

`https://[account-hostname]/et/`

Here is a complete example:

```
<img src="https://account.uxmail.io/et/?et_products=1&et_currency=USD&et_customer_
↪ref=12345678&et_order_ref=12345678&et_prod_0_id=54491472&et_prod_0_quantity=1&et_
↪prod_0_name=Coca+Cola+500ml&et_prod_0_category=Softdrinks&et_prod_0_unit_price=1.50
↪" height="1" width="1">
```

Anti-tampering

Since this mode of transferring order data happens on the client side of the transaction you are leaving your order data open for examination and possibly tampering in transit.

You may configure your account with an anti-tampering key, you and digitally sign the parameters to at least avoid anyone changing them in transit. Use the following procedure:

1. Take the complete URL encoded parameter list (e.g. `bar=baz&foo=quux` sorted by the key name.
2. Feed this string into an HMAC-SHA256 keyed by the anti-tampering key from your account.

3. Hex encode the resulting digest and add as the `et_digest` parameter.

Here's the above example with the added `et_digest` parameter:

```
<img src="https://account.uxmail.io/et/?et_currency=USD&et_customer_ref=12345678&et_
↪order_ref=12345678&et_prod_0_category=Softdrinks&et_prod_0_id=54491472&et_prod_0_
↪name=Coca+Cola+500ml&et_prod_0_quantity=1&et_prod_0_unit_price=1.50&et_products=1&
↪et_digest=..." height="1" width="1">
```

Warning: You must calculate the HMAC-SHA256 digest server side for the digest to have any effect. Otherwise your secret key would be compromised.

Debugging

If results fail to show up in the statistics, there might be some problems with registering the sales. In case of an error, the resulting 1x1 pixel tracking GIF will contain an extra HTTP header called `ST-Error` with any errors encountered during sales registration.

CHAPTER

SIX

WEBHOOKS

Webhooks is an easy way of integrating your existing systems with Ubivox. Webhooks are just a single HTTP POST request sent for every specific event type.

6.1 Request structure

Webhook messages are sent as HTTP POST requests to the URL specificed. Messages are sent asynchronously to the trigger event, and may be delayed. Messages may not arrive in order. All messages consists of the following HTTP POST variables:

- `time` - Local time when the event occurred. E.g.: `2009-12-31T23:59:59`
- `data` - The JSON encoded contents of the message

6.2 Throttling

When dealing with large systems, the amount of webhook calls may slow your systems down. Ubivox deals with this by allowing you to set a throttle parameter for each webhook. This defines a maximum numbers of webhook messages per minute for the particular webhook Ubivox will try to deliver.

6.3 Delivery guarantee

In order to make sure that each message arrives securely at the recieving server, we monitor the HTTP status codes in the replies. If we are given anything else than a `2xx` reply, we will retry the current message later. Messages are retried at an interval of at least five minutes until the retry limit has been hit.

6.4 Setting up

Webhooks are setup through the Ubivox interface by choosing *Account* then *Webhooks* in the top menu.

6.5 Supported webhooks

We currently support webhooks for the following events:

6.5.1 subscription

Subscription event

```
{
    "address": "example@example.org",
    "list": "Company Contacts",
    "list_id": 5,
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    }
}
```

6.5.2 updated_email

Updated e-mail address

```
{
    "new_address": "new-example@example.org",
    "old_address": "old-example@example.org"
}
```

6.5.3 unsubscription

Unsubscription event (by subscriber request)

```
{
    "address": "example@example.org",
    "list": "Company Contacts",
    "list_id": 5,
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    }
}
```

6.5.4 remove_subscription

Unsubscription event (by administrator)

```
{
    "address": "example@example.org",
    "list": "Company Contacts",
    "list_id": 5,
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    }
}
```

6.5.5 suspend_subscription

Suspend event (due to delivery failure)

```
{
    "address": "example@example.org",
    "list": "Company Contacts",
    "list_id": 5,
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    }
}
```

6.5.6 activate_subscription

Reactivation of a suspended subscription

```
{
    "address": "example@example.org",
    "list": "Company Contacts",
    "list_id": 5,
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    }
}
```

6.5.7 login

Login event (in the administrative interface)

```
{
    "from": "208.77.188.166",
    "username": "example"
}
```

6.5.8 junk_delivery

Junked delivery event

```
{
    "address": "example@example.org",
    "delivery_id": 10,
    "list": "Company Contacts",
    "list_id": 5,
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

6.5.9 junk_email

Junked e-mail event (from E-mail API)

```
{
    "reference": "..."
}
```

6.5.10 bounced_email

Bounced e-mail event (from E-mail API)

```
{
    "address": "example@example.org",
    "classification": "hard|soft",
    "reference": "..."
}
```

6.5.11 delivery_started

Delivery of a newsletter has started

```
{
    "delivery_id": 42,
    "delivery_subject": "My newsletter",
    "list": "Company Contacts",
    "list_id": 5
}
```

6.5.12 delivery_read

First read of a delivery by a subscriber

```
{
    "address": "example@example.org",
    "delivery_id": 42,
    "delivery_subject": "My newsletter",
    "event": "open",
    "list": "Company Contacts",
    "list_id": 5,
    "remote_addr": "198.51.100.60",
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    },
```

(continues on next page)

(continued from previous page)

```
    "user_agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like␣
↪Gecko) Chrome/41.0.2228.0 Safari/537.36"
}
```

6.5.13 `link_clicked`

First click of a link by a subscriber

```
{
    "address": "example@example.org",
    "delivery_id": 42,
    "delivery_subject": "My newsletter",
    "event": "click",
    "link_id": 80,
    "link_location": "https://www.example.org/",
    "list": "Company Contacts",
    "list_id": 5,
    "remote_addr": "198.51.100.60",
    "subscriber_data": {
        "City": "Springfield",
        "Firstname": "John",
        "Lastname": "Doe"
    },
    "user_agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like␣
↪Gecko) Chrome/41.0.2228.0 Safari/537.36"
}
```

CHAPTER

SEVEN

POST HANDLER

This chapter documents the HTTP POST form handler for implementing your own custom HTML forms on your website.

7.1 Introduction

Using the HTTP POST form handler, you can subscribe and unsubscribe users to your newsletters in a very easy way. You will do all the configuration with regards to redirection in hidden fields of the POST request.

7.2 Form fields

These are the required form fields you will need for the request to succeed:

Name	Description
`action`	Must be either `subscribe` or `unsubscribe`
`email_address`	The e-mail address of the subscriber to be either subscribed or unsubscribed
`lists`	A comma-separated list of Maillist IDs which the request deals with.

7.2.1 Subscriber data fields

If you need to save data fields along with the subscriptions, you must send additional fields named after the data field keys prefixed with `data_`. E.g: `data_Name`.

7.2.2 Multiple choice

If you need to supply multiple options for a single data field value, repeat the input field with the same name.

7.3 Example HTML forms

Below is some simple examples of HTTP POST forms. Care must be taken to examine the errors parameter by the server (either server-side og client-side using javascript).

7.3.1 Subscribe

This form will subscribe the user to lists 1 and 2 and update the subscriber data with Name and the multiple choice data field Interests:

```
<!-- Example HTML form -->

<form action="https://account.clients.ubivox.com/handlers/post/" method="post">

  <input type="hidden" name="action" value="subscribe" />
  <input type="hidden" name="lists" value="1,2" />

  <p>
    <label for="email_address_id">E-mail address</label>
    <input type="text" name="email_address" id="email_address_id" />;
  </p>

  <p>

    <!-- Example single text input field -->

    <label for="data_Name_id">Name</label>
    <input type="text" name="data_Name" id="data_Name_id" />;

  </p>

  <p>

    <!-- Example multiple choice field -->

    <label>Interests</label><br>

    <input type="checkbox" name="data_Interests" value="Music" id="data_Interests_0_id
↪" />
    <label for="data_Interests_0_id">Music</label><br>

    <input type="checkbox" name="data_Interests" value="Food" id="data_Interests_0_id
↪" />
    <label for="data_Interests_0_id">Food</label><br>

    <input type="checkbox" name="data_Interests" value="Sports" id="data_Interests_0_
↪id" />
    <label for="data_Interests_0_id">Sports</label><br>

  </p>

  <p>
    <input type="submit" value="Subscribe" />
  </p>

</form>
```

7.3.2 Unsubscribe

This form will unsubscribe the user from lists 1 and 2:

```
<!-- Example HTML form -->

<form action="https://account.clients.ubivox.com/handlers/post/" method="post">

  <input type="hidden" name="action" value="unsubscribe" />
  <input type="hidden" name="lists" value="1,2" />

  <p>
    <label for="email_address_id">E-mail address</label>
    <input type="text" name="email_address" id="email_address_id" />;
  </p>

  <p>
    <input type="submit" value="Unsubscribe" />
  </p>

</form>
```

7.4 Your own status pages

If you want to use your own status pages instead of the standard ones, you can send the following two parameters as hidden fields in your form:

- **`success_url`** The URL to which the system will redirect the user after a successful operation
- **`failure_url`** The URL to which the system will redirect the user after a failed operation. (See *Error handling* below)

7.5 Error handling

The form system operates with two kinds of errors. Fatal errors and normal errors. Fatal errors should only occur during the development and testing of your forms. These errors will halt the subscription process and produce an error message on your screen. These consists of:

- Missing 'action' parameter.
- Missing 'lists' parameter.
- Missing 'email_address' parameter.
- Unknown data key.
- Uknown mailing list.
- Opt-in not configured yet.

The other set of errors is errors that requires the end user to take action. If you use your own `failure_url` status page, any errors occurred during the handling process, will be sent to the page as a set of HTTP GET parameter called `errors` (textual representation) and `error_codes` (numerical error codes) - both separated by `;`.

If you haven't configured your own status page, the system will use a set of standard templates to present the user with error messages and choices. Currently, these are the normal errors that can occur:

- Missing required data key (Error code: 10)
- E-mail address is already subscribed (Error code: 11)

- E-mail address is not valid (Error code: 12)
- Unknown E-mail address (Error code: 13)
- E-mail address was not subscribed (Error code: 14)
- E-mail address on stop list (Error code: 15)

Note: You should always rely on the error code and not use the textual representations for anything other than debugging. Otherwise you may risk leaving your system open for XSS or phising attacks.

7.5.1 Example parameters for custom error pages

Here is an example using custom error pages:

```
<input type="hidden" name="success_url" value="http://www.example.com/ok/" />
<input type="hidden" name="failure_url" value="http://www.example.com/error/" />
```

7.6 Internationalization

All messages and notifications when using the HTTP POST handler, can be internationalized to suit the language of your website. We currently support these languages:

- `da`: Danish
- `de`: German
- `en`: English
- `es`: Spanish
- `fr`: French
- `it`: Italian
- `nb`: Norwegian
- `nl`: Dutch
- `sv`: Swedish

The language code must be inserted as a hidden field in your HTML form called `language_code`. Example:

```
<input type="hidden" name="language_code" value="es" />
```

If no language code is given, The system will try to determine the users preferred language depending on their web browser settings.

7.7 Opt-in scheme

In special circumstances it might be required that the system does not send the confirmation e-mail to new subscribers. To use this feature, approval from your support team is **required**.

Warning: Having a public web form with no confirmation for new subscriptions leaves you open to subscription bombing with a certain loss of reputation for your sending identity and possibly your company name.

Controlling the confirmation e-mail is done using the `optin_scheme` form field.

7.7.1 Double

To use double opt-in (with confirmation e-mail):

```
<input type="hidden" name="optin_scheme" value="double" />
```

This is the default and recommended behaviour.

7.7.2 Single

To use single opt-in (**no** confirmation e-mail)

```
<input type="hidden" name="optin_scheme" value="single" />
```

This option requires prior approval from your support team.

CHAPTER

EIGHT

FEEDS

Ubivox gives you easy access to your data using a variety of feeds in different formats. This information can be used for analyzing data in Excel or for integrating with external systems.

8.1 Supported formats

- `json` - JSON encoded
- `csv` - CSV encoded (separated by `;`)

The format must be passed as the `(format)` parameter in the URL.

8.2 Other feed parameters

Some feeds require other parameters. These must be replaced in the URL before use. They are denoted by the `(parameter)` syntax below.

8.3 Base URL

You should use your Ubivox account URL as the base URL for all requests to the feeds.

8.4 Example

If the feeds documentation lists the following URL:

```
GET /feeds/delivery_link_used_by/(format)/delivery_id=(delivery_id)/
link_id=(link_id)/
```

The actual URL for the JSON feed of delivery ID 283 and link ID 9454 will be:

```
https://account.clients.ubivox.com/feeds/delivery_link_used_by/json/
delivery_id=283/link_id=9454/
```

8.5 Available feeds

8.5.1 `archived` feeds

`GET /feeds/archived_deliveries/` (*format*) **`/`**
Retrieves your archived deliveries.

Request Headers

- Authorization – Not required

Status Codes

- 200 OK – No error

`GET /feeds/archived_deliveries_list/` (*format*) **`/list_id=`**
*list_id***`/`** Retrieves your archived deliveries for a given list.

Request Headers

- Authorization – Not required

Status Codes

- 200 OK – No error

8.5.2 `client` feeds

`GET /feeds/client_statistics/` (*format*) **`/`**
Retrieves some global statistics for your account.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.3 `contacts` feeds

`GET /feeds/contacts/` (*format*) **`/`**
Contact count for your account

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.4 `course` feeds

GET /feeds/course_data/ (*format*) **/list_id=**
*list_id***/stage_id=***stage_id***/period=***period***/** Retrieves statistical data about courses.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.5 `data` feeds

GET /feeds/data_fields/ (*format*) **/**
Retrieves data about your subscriber data fields.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.6 `deliveries` feeds

GET /feeds/deliveries/ (*format*) **/**
Retrieves data about your deliveries including some statistics.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.7 `delivery` feeds

GET /feeds/delivery/ (*format*) **/delivery_id=**
*delivery_id***/** Retrieves information for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_bounce_rate_breakdown/ (*format*) **/delivery_id=**
*delivery_id***/** Retrieves a bounce rate breakdown in percent for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_bounces/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a list of bounces for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_complaints/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a list of complaints for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_devices/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a list of opens per device for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_hour_interval_statistics/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a list of hour intervals with view and click counts for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_link_used_by/ (*format*) **/delivery_id=***delivery_id***/link_id=***link_id***/** Retrieves a feed of the subscribers who has used a given link

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_links/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a link summary for a delivery

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_locations_countries/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a list of opens per country for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_locations_regions/ (*format*) **/delivery_id=***delivery_id***/country_code=***country_code***/** Retrieves a list of opens per region for a given country on a delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_rates/ (*format*) **/delivery_id=***delivery_id***/** Retrieves view and click rate for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_reads/ (*format*) **/delivery_id=***delivery_id***/** Retrieves a list of recipients that had a read registered for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_recipients/ (*format*) **/delivery_id=**
*delivery_id***/** Retrieves a list of recipients for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_target_sales/ (*format*) **/delivery_id=**
*delivery_id***/target_id=***target_id***/** Retrieves a feed of the sales tracked by a given target for a given delivery

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_unsubscriptions/ (*format*) **/delivery_id=**
*delivery_id***/** Retrieves a list of unsubscription for a given delivery.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_view_link_statistics/ (*format*) **/delivery_id=**
*delivery_id***/** Retrieves view and link statistics for a given delivery. (10 days)

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/delivery_view_link_statistics_full/ (*format*) **/delivery_id=**
*delivery_id***/** Retrieves view and link statistics for a given delivery. (Full)

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.8 `email` feeds

GET /feeds/email_hour/ (*format*) **/smtpuser_id=**
*smtpuser_id***/** Retrieves hourly usage for the last 90 days for transaction e-mails.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/email_month/ (*format*) **/smtpuser_id=**
*smtpuser_id***/** Retrieves usage per day for transaction e-mails.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/email_year/ (*format*) **/smtpuser_id=**
*smtpuser_id***/** Retrieves usage per month for transaction e-mails.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.9 `global` feeds

GET /feeds/global_delivery_statistics/ (*format*) **/**
Retrieves global statistics for your deliveries for your account.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/global_subscription_growth/ (*format*) **/**
Retrieves global statistics for your subscription growth for your account.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error

- 401 Unauthorized – Missing or wrong credentials supplied

8.5.10 `list` feeds

GET /feeds/list_deliveries/ (*format*) **/list_id=**
*list_id***/** Retrieves data about your deliveries for a given list including some statistics.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_hour_interval_statistics/ (*format*) **/list_id=**
*list_id***/** Retrieves data about your most active hours of the day.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_info/ (*format*) **/list_id=**
*list_id***/** Retrieves information about one of your lists.

Request Headers

- Authorization – Not required

Status Codes

- 200 OK – No error

GET /feeds/list_primary_data_fields/ (*format*) **/list_id=**
*list_id***/** Retrieves information about the primary, public data fields of one of your lists.

Request Headers

- Authorization – Not required

Status Codes

- 200 OK – No error

GET /feeds/list_signins/ (*format*) **/list_id=**
*list_id***/** Retrieves the last 10 signins to a given list.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_signouts/ (*format*) **/list_id=**
*list_id***/** Retrieves the last 10 signouts from a given list.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_growth/ (*format*) **/list_id=***list_id***/** Retrieves data about your subscription growth for a given list in the past 10 days.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_growth_180days/ (*format*) **/list_id=***list_id***/** Retrieves data about your subscription growth for a given list in the past 180 days.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_growth_30days/ (*format*) **/list_id=***list_id***/** Retrieves data about your subscription growth for a given list in the past 30 days.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_growth_365days/ (*format*) **/list_id=***list_id***/** Retrieves data about your subscription growth for a given list in the past 365 days.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_growth_730days/ (*format*) **/list_id=***list_id***/** Retrieves data about your subscription growth for a given list in the past 730 days.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_growth_90days/ (*format*) **/list_id=** *list_id***/** Retrieves data about your subscription growth for a given list in the past 90 days.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/list_subscription_references/ (*format*) **/list_id=** *list_id***/** Retrieves data about your subscription references.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.11 `lists` feeds

GET /feeds/lists/ (*format*) **/**
Retrieves data about your lists on Ubivox with title and subscription counts.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

8.5.12 `partner` feeds

GET /feeds/partner_expiring_clients/ (*format*) **/**
Retrieves a feed of expiring clients

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/partner_latest_deliveries/ (*format*) **/**
Retrieves a feed of the latest deliveries

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/partner_latest_signups/ (*format*) **/**
Retrieves a feed of the latest account signups

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

GET /feeds/partner_status/ (*format*) **/**
Retrieves a status feed of new clients, sales etc.

Request Headers

- Authorization – Requires HTTP Basic authentication using the feeds credentials

Status Codes

- 200 OK – No error
- 401 Unauthorized – Missing or wrong credentials supplied

CHAPTER

NINE

EXAMPLES

In order to get you started with using the API, we have collected some examples in a couple of different languages for you.

9.1 Python Examples

Python really shines when it comes to XML-RPC support. The standard library has the excellent `xmlrpclib` bundled and it will work perfectly with the Ubivox API.

Helpful links for Python

Python Library Documentation for `xmlrpclib`: http://docs.python.org/library/xmlrpclib.html

Python Module of the Week: xmlrpclib: http://www.doughellmann.com/PyMOTW/xmlrpclib/

9.1.1 Creating a new subscription

```
from xmlrpclib import ServerProxy

server = ServerProxy("https://username:password@company.clients.ubivox.com/xmlrpc/")

# Using opt-in to list ID 42 for user@example.com
server.ubivox.create_subscription("user@example.com", [42], True)

# Not using opt-in to list ID 42 for user@example.com
server.ubivox.create_subscription("user@example.com", [42], False)

# Not using opt-in to list ID 42 and 60 for user@example.com
server.ubivox.create_subscription("user@example.com", [42, 60], False)
```

9.1.2 Updating subscriber data

```
from xmlrpclib import ServerProxy

server = ServerProxy("https://username:password@company.clients.ubivox.com/xmlrpc/")

# Set the data field Name for user@example.com
server.ubivox.set_subscriber_data("user@example.com", {"Name": "John Doe"})
```

9.1.3 Unsubscribing

```
from xmlrpclib import ServerProxy

server = ServerProxy("https://username:password@company.clients.ubivox.com/xmlrpc/")

# Cancel a subscription to list ID 42 for user@example.com
server.ubivox.cancel_subscription("user@example.com", [42])

# Cancel a subscription to list ID 42 and 60 for user@example.com
server.ubivox.cancel_subscription("user@example.com", [42, 60])
```

9.1.4 Create and send a delivery

```
import datetime

from xmlrpclib import ServerProxy

server = ServerProxy("https://username:password@company.clients.ubivox.com/xmlrpc/")

# Create a new delivery on list ID 42
delivery_id = server.ubivox.create_delivery(
    "My first newsletter",
    "HTML body",
    "Text body",
    42
)

# Schedule the delivery for delivery ASAP
now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
server.ubivox.send_delivery(delivery_id, now)
```

9.1.5 Error/Exception handling

```
from xmlrpclib import ServerProxy, Fault

server = ServerProxy("https://username:password@company.clients.ubivox.com/xmlrpc/")

try:

    # Using opt-in to list ID 42 for user@example.com
    server.ubivox.create_subscription("user@example.com", [42], True)

except Fault, e:

    # You can use your own error message, by checking the e.faultCode parameter

    if e.faultCode == 1003:
        print "You are already subscribed"

    # Or use the one Ubivox supplies for you, available in e.faultString

    print e.faultString
```

(continues on next page)

(continued from previous page)

```
    # This will print "The user is already subscribed" if that was the error.
else:
    # No exception happened: Everything went well!
    print "You are now subscribed."
```

9.2 PHP Examples

Our PHP examples utilize our Ubivox API PHP client to communicate with the Ubivox API.

More information about our PHP client

Our client is hosted in Bitbucket, where you can download and report bugs as well: https://bitbucket.org/ubivox/ubivox-api-php

9.2.1 Creating a new subscription

```
<?php

require_once "ubivox_api.php";

$client = new UbivoxAPI(
    "username",
    "password",
    "https://company.clients.ubivox.com/xmlrpc/"
);

// Using opt-in to list ID 42 for user@example.com
$client->call("ubivox.create_subscription",
              array("user@example.com", 42, true));

// Not using opt-in to list ID 42 for user@example.com
$client->call("ubivox.create_subscription",
              array("user@example.com", 42, false));

// Not using opt-in to list ID 42 and 60 for user@example.com
$client->call("ubivox.create_subscription",
              array("user@example.com", array(42, 60), false));

?>
```

9.2.2 Updating subscriber data

```
<?php
```

(continues on next page)

(continued from previous page)

```php
require_once "ubivox_api.php";

$client = new UbivoxAPI(
    "username",
    "password",
    "https://company.clients.ubivox.com/xmlrpc/"
);

// Set the data field Name for user@example.com
$client->call("ubivox.set_subscriber_data",  array(
    "user@example.com",
    array("Name" => "John Doe")
));

?>
```

9.2.3 Unsubscribing

```php
<?php

require_once "ubivox_api.php";

$client = new UbivoxAPI(
    "username",
    "password",
    "https://company.clients.ubivox.com/xmlrpc/"
);

// Cancel a subscription to list ID 42 for user@example.com
$client->call("ubivox.cancel_subscription",
              array("user@example.com", 42));

// Cancel a subscription to list ID 42 and 60 for user@example.com
$client->call("ubivox.cancel_subscription",
              array("user@example.com", array(42, 60)));

?>
```

9.2.4 Create and send a delivery

```php
<?php

require_once "ubivox_api.php";

$client = new UbivoxAPI(
    "username",
    "password",
    "https://company.clients.ubivox.com/xmlrpc/"
);

// Create a new delivery on list ID 42
$delivery_id = $client->call("ubivox.create_delivery", array(
```

(continues on next page)

(continued from previous page)

```
    "My first newsletter",
    "HTML body",
    "Text body",
    42
));

// Schedule the delivery for delivery ASAP
$now = date("Y-m-d H:i:s");

$client->call("ubivox.send_delivery", array($delivery_id, $now));

?>
```

9.2.5 Error/Exception handling

```
<?php

require_once "ubivox_api.php";

$client = new UbivoxAPI(
    "username",
    "password",
    "https://company.clients.ubivox.com/xmlrpc/"
);

try {

  // Using opt-in to list ID 42 for user@example.com
  $client->call("ubivox.create_subscription",
                array("user@example.com", 42, 1));

} catch(UbivoxAPIError $e) {

  // You can use your own error message, by checking the $e->getCode() parameter
  if ($e->getCode() == 1003) {
          print "You are already subscribed";
  }

  // Or use the one Ubivox supplies for you, available in $e->getMessage()
  print $e->getMessage();

  // This will print "The user is already subscribed" if that was the error.

}

// No exception happened: Everything went well!
print "You are now subscribed.";

?>
```

9.3 Node.js Examples

The Node.js examples here use the `xmlrpc` NPM package:

Links for using XML-RPC with Node.js

The `xmlrpc` NPM package https://www.npmjs.com/package/xmlrpc

How to use NPM to install the `xmlrpc` package https://www.npmjs.com/package/xmlrpc/tutorial

The `moment` NPM package for date/time handling https://www.npmjs.com/package/moment

9.3.1 Creating a new subscription

```
var xmlrpc = require("xmlrpc");

var client = xmlrpc.createSecureClient({
    host: "company.clients.ubivox.com",
    path: "/xmlrpc/",
    basic_auth: {
        user: "username",
        pass: "password"
    }
});

// Using opt-in to list ID 42 for user@example.com
client.methodCall(
    "ubivox.create_subscription",
    ["user@example.com", 42, true],
    function (error, value) {
        if (error) throw error;
    }
);

// Not using opt-in to list ID 42 for user@example.com
client.methodCall(
    "ubivox.create_subscription",
    ["user@example.com", 42, false],
    function (error, value) {
        if (error) throw error;
    }
);

// Not using opt-in to list ID 42 and 60 for user@example.com
client.methodCall(
    "ubivox.create_subscription",
    ["user@example.com", [42, 60], false],
    function (error, value) {
        if (error) throw error;
    }
);
```

9.3.2 Updating subscriber data

```
var xmlrpc = require("xmlrpc");

var client = xmlrpc.createSecureClient({
```

(continues on next page)

(continued from previous page)

```
    host: "company.clients.ubivox.com",
    path: "/xmlrpc/",
    basic_auth: {
        user: "username",
        pass: "password"
    }
});

// Set the data field Name for user@example.com
client.methodCall(
    "ubivox.set_subscriber_data",
    ["user@example.com", {
        Name: "John Doe"
    }],
    function (error, value) {
        if (error) throw error;
    }
);
```

9.3.3 Unsubscribing

```
var xmlrpc = require("xmlrpc");

var client = xmlrpc.createSecureClient({
    host: "company.clients.ubivox.com",
    path: "/xmlrpc/",
    basic_auth: {
        user: "username",
        pass: "password"
    }
});

// Cancel a subscription to list ID 42 for user@example.com
client.methodCall(
    "ubivox.cancel_subscription",
    ["user@example.com", 42],
    function (error, value) {
        if (error) throw error;
    }
);

// Cancel a subscription to list ID 42 and 60 for user@example.com
client.methodCall(
    "ubivox.cancel_subscription",
    ["user@example.com", 42, 60],
    function (error, value) {
        if (error) throw error;
    }
);
```

9.3.4 Create and send a delivery

```
var xmlrpc = require("xmlrpc");
var moment = require("moment");

var client = xmlrpc.createSecureClient({
    host: "company.clients.ubivox.com",
    path: "/xmlrpc/",
    basic_auth: {
        user: "username",
        pass: "password"
    }
});

// Create a new delivery on list ID 42
client.methodCall(

    "ubivox.create_delivery",
    ["My first newsletter", "HTML body", "Text body", 42],

    function (error, value) {

        if (error) throw error;

        var delivery_id = value;

        // Schedule the delivery for delivery ASAP
        var now = moment().format("YYYY-MM-DD HH:mm:ss");

        client.methodCall(
            "ubivox.send_delivery",
            [delivery_id, now],
            function (error, value) {
                if (error) throw error;
            }
        );

    }
);
```

9.3.5 Error/Exception handling

```
var xmlrpc = require("xmlrpc");

var client = xmlrpc.createSecureClient({
    host: "company.clients.ubivox.com",
    path: "/xmlrpc/",
    basic_auth: {
        user: "username",
        pass: "password"
    }
});

// Using opt-in to list ID 42 for user@example.com
client.methodCall(
```

(continues on next page)

(continued from previous page)

```
    "ubivox.create_subscription",
    ["user@example.com", 42, true],
    function (error, value) {

        if (error) {

            if (error.faultCode == 1003) {
                console.log("You are already subscribed");
            } else {
                console.log(error.faultString);
            }

        } else {

            console.log("You are now subscribed");

        }

    }
);
```

CHAPTER

TEN

CHANGELOG

We will document changes to the API in this chapter. We try to not introduce backwards incompatible changes unless they cannot be avoided and it would cause errors to leave them out.

Only releases with API changes will be noted, which is why the list below appears to skip releases.

10.1 2.34.0 - Sep 20, 2018

- New method to check for active subscription on list: *mailer.has_active_subscription()*.

10.2 2.33.0 - May 3, 2018

- New method to support the Right To Be Forgotten: *mailer.forget_subscriber()*.

10.3 2.31.0 - July 17, 2017

- New methods for controlling sender details on newsletters: *mailer.set_delivery_sender()*, *mailer.set_delivery_sender_rewrite_data_field()*.
- The above method also required new error code `5022` for invalid sender configurations.
- New methods for accessing consent details for subscriptions: *mailer.get_subscription_consent_details()*, *mailer.set_subscription_consent_details()*.

10.4 2.30.1 - May 23, 2017

- New Ecommerce Tracking order parameter `et_added` for order time. See *Ecommerce Tracking*.
- New method for looking up error code `1001` causes for *subscription methods*: *mailer.create_subscription_error()*

10.5 2.30.0 - April 6, 2017

- Support for *Ecommerce Tracking*.
- New error codes for *mailer.send_email()* and *mailer.send_email_with_names()*: `5021`.

10.6 2.29.0 - February 2, 2017

- Multiple choice support in *POST Handler*

10.7 2.28.0 - January 19, 2017

- Changed the XML-RPC namespace from `ubivox` to `mailer`. The old `ubivox` namespace will continue to work.
- Moved feeds documentation here: *Feeds*

10.8 2.27.1 - October 6, 2016

- Moved web form documentation here: *POST Handler*

10.9 2.23.0 - December 27, 2015

- Support for XML-RPC multicalls. See *system.multicall()*

10.10 2.22.1 - November 24, 2015

- New API functions for working with delivery templates: `mailer.create_delivery_from_template()`, *mailer.update_delivery_template_content()*, *mailer.get_delivery_template_content()*

10.11 2.22.0 - October 27, 2015

- New error code for handling encoding errors in RFC2822 formatted e-mails: `5019` in *mailer.send_email_rfc2822()*.

10.12 2.21.4 - October 22, 2015

- New API methods for invoice access: *mailer.account_invoice_list()*, *mailer.account_invoice_pdf()*, *mailer.account_invoice_details()*

10.13 2.21.3 - October 8, 2015

- Correct handling of archived lists in API. Affected methods has had the `2005` fault introduced for invalid operations on an archived list.
- New error code for dealing with blocked accounts receiving new subscription `1011`.

10.14 2.20.0 - September 3, 2015

- New API methods: *mailer.delete_webhook_by_url()*,

10.15 2.19.0 - July 30, 2015

- New error code for *mailer.set_subscriber_data()*, *mailer.stop_subscriber()* when sending invalid e-mail addresses: `1001`

10.16 2.17.0 - May 12, 2015

- New *splittest methods*
- New API methods for administrative unsubscriptions: *mailer.remove_subscription()*, *mailer.remove_all_subscriptions()*,

10.17 2.16.0 - March 12, 2015

- New webhooks: *delivery_read*, *link_clicked*.

10.18 2.15.1 - January 20, 2015

- New *exports methods*

10.19 2.14.2 - September 9, 2014

- New API method *mailer.set_delivery_in_archive()*.

10.20 2.13.10 - August 12, 2014

- New API method *mailer.list_subscriptions()*.
- Added `activated`, renamed `deleted` to `suspended` in return struct of *mailer.get_subscriber()*.

10.21 2.13.7 - May 6, 2014

- New API method: *mailer.update_subscriber_email()*.

10.22 2.13.0 - January 27, 2014

- New API method: *mailer.copy_delivery()*.
- New API method: *mailer.update_data_field_options()*.
- Webhooks documentation added: *Webhooks*.

10.23 2.11.6 - November 19, 2013

- New API method: *mailer.test_delivery()*,
- Introduced extra statistics in *mailer.get_delivery_stats()*,
- Fixed a bug in *mailer.synchronize()* that would result in some unsubscriptions missing.

10.24 2.11.3 - October 22, 2013

- New API methods: *mailer.get_stopped_subscribers_since()*, *mailer.get_maillist()*,

10.25 2.11.1 - October 1, 2013

- New API methods: *mailer.stop_subscriber()*, *mailer.get_stopped_subscribers()*, *mailer.maillist_archive_meta()*,

10.26 2.11.0 - September 11, 2013

- New API methods: *mailer.account_flag()*, *mailer.get_subscriber_delivery_stats()*, *mailer.synchronize()*, *mailer.time_zone()*, *partner.sudo()*

10.27 2.10.6 - August 20, 2013

- Updated PHP examples with new API client: *PHP Examples*.

10.28 2.10.5 - August 13, 2013

- New method: *mailer.copy_maillist()*.

10.29 2.10.3 - July 18, 2013

- New error code for *Partner API Methods*: `7009`.
- Ubivox logos added.

10.30 2.10.1 - June 25, 2013

- Documented our new XML-RPC debugging facility. See *Debugging*.

10.31 2.10.0 - June 19, 2013

- New error codes for *email methods*: `9995`.

10.32 2.9.6 - May 27, 2013

- Added `recipients` and `delivered` to *`mailer.account_status()`* (on published newsletters).

10.33 2.9.3 - April 22, 2013

- New webhook API, see *webhook methods*.

10.34 2.9.2 - March 26, 2013

- Added new method for accessing the list archive from the API: *`mailer.maillist_archive()`*. Also added the archive HTML version of the newsletter to *`mailer.get_delivery()`*.
- Added choices and sort order to *`mailer.list_data_fields_details()`*.
- Added new call: *`mailer.account_status()`*.

10.35 2.9.1 - March 11, 2013

- Added support for targets: *`mailer.create_target()`*, *`mailer.list_targets()`*, *`mailer.update_target()`* and *`mailer.target_statistics()`*.

10.36 2.9.0 - March 6, 2013

- Added send_time to results for calls to *`mailer.get_delivery()`*, *`mailer.list_deliveries()`*, and *`mailer.list_all_deliveries()`*.
- Corrected bug in *`mailer.create_data_field()`* and *`mailer.update_data_field()`*
- Added *`mailer.time_zones()`* to expose the supported time zones.
- Added *`partner.available_account_types()`* and *`partner.client_details()`* to *Partner API Methods*.

10.37 2.8.1 - December 18, 2012

- Added clarification regarding *Data fields with predefined options*.

10.38 2.7.2 - October 24, 2012

- Added new chapter about our new Partner API that enables Ubivox Partner to mechanically create and update client accounts in Mailer. See *Partner API Methods*.

10.39 2.7.0 - October 11, 2012

- Added clarification of the data type of the `data` parameter in `mailer.import_new()`.
- Added new chapter about our new Sales Tracking API that enables you to track the revenue generated by your newsletters. See *Sales tracking*.
- Added new error condition to `mailer.create_delivery()` and `mailer.update_delivery()`: `3004` used to indicate a problem with the delivery content.
- Added support for the time zone configured for your Ubivox account. See *Date/time formats and time zone*.
- Added clarification of the subscriber data structure. See *Subscriber data*.

10.40 2.5.0 - June 14, 2012

- New authentication feature: Multiple API logins, see *Authentication*.
- Corrected error in import API. `mailer.import_new()` does not take an *overwrite* argument.

10.41 2.4.2 - April 19, 2012

- New import API, see *imports methods*.

10.42 2.4.0 - March 15, 2012

- `mailer.list_data_fields()` and `mailer.get_data_keys()` now actually returns the key and not the title of the data field.
- `mailer.create_data_field()`: Has a new argument: *key*.
- `mailer.update_data_field()`: Changed *old_title* argument to *key*.
- Changed the error *InvalidDataTitle* to *InvalidDataKey*. Still has the error code 1005.

CHAPTER

ELEVEN

QUESTIONS

If you run into any problems using the API, do not hesitate to contact our support team standing by at **support@ubivox.com**.

/feeds

GET /feeds/partner_expiring_clients/(format)/, 80
GET /feeds/partner_latest_deliveries/(format)/, 80
GET /feeds/partner_latest_signups/(format)/, 81
GET /feeds/partner_status/(format)/, 81

INDEX