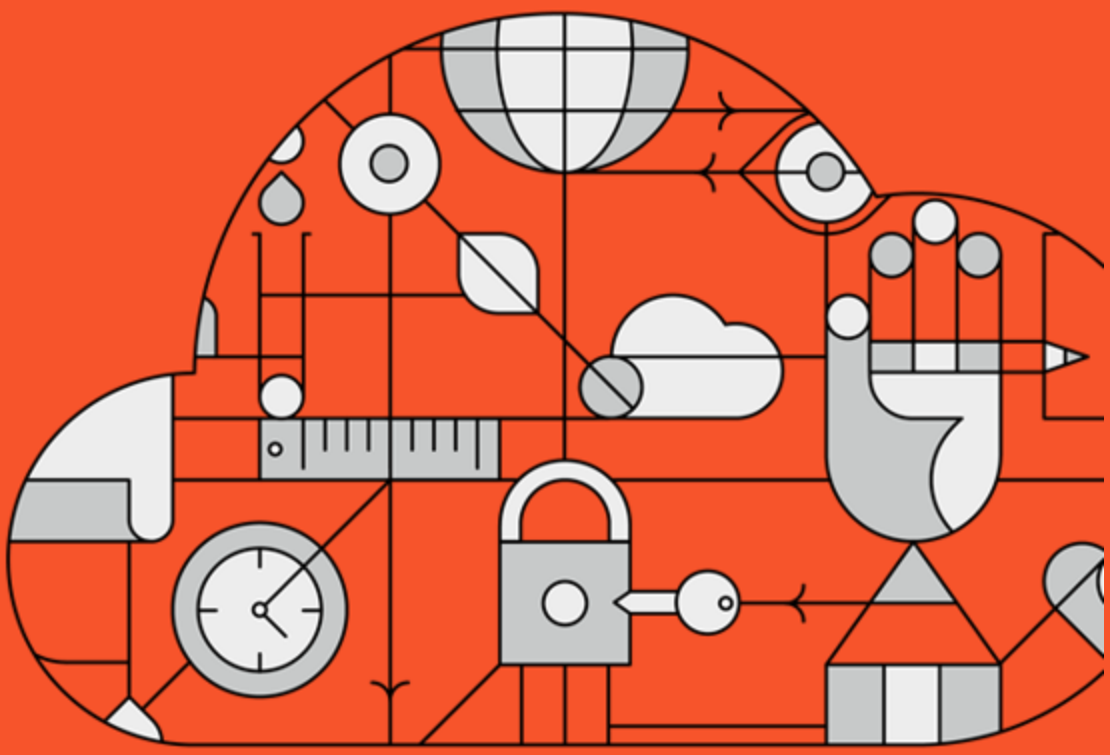
Episerver Campaign

HTTP API



Digital Experience Delivered.

Episerver Campaign HTTP API

Version 6.70

Table of contents

HTTP API

The HTTP API of Episerver Campaign is an easy to configure and use interface, which can be used to execute the most common functions with an HTTP request. If a user clicks on a button or a link on a website or in an email (e.g. **Send registration**), a specific URL is called and the parameters are submitted. This URL contains an action to be executed (e.g. **sendEventMail**) and further parameters (e.g. recipient ID, mailing ID). The API server reads these parameters and calls the requested action.

What can I do with the HTTP API?

Use the HTTP API to integrate online forms, for example registration forms for new subscribers. It it also suitable to sent triggered emails, such as order confirmations or invoices. The following functions can be used with the HTTP API:

- Create new recipients
- Send triggered emails
- Blacklist recipients
- Unsubscribe recipients

Modifications since the previous version

The **bmOptinFrequencyLimit** parameter has been added to the **subscribe** operation .

Introduction to the HTTP API

The following documentation describes the HTTP mail service and form service for Episerver Campaign.

Application possibilities

The email marketing solution Episerver Campaign cannot only be used with the graphical user interface (GUI). With the help of an HTTP interface, frequently used functions can be integrated into your own website. In addition, functions can be performed by email.

The HTTP-API provides a simple method to integrate Episerver Campaign in your website. To perform functions, you simply call a URL and pass some parameters.

Additionally, the HTTP-API provides special URLs to add data to recipient profiles by using emails.

The HTTP-API provides the following features:

- All functions are executed using a simple HTTP request (GET or POST).
- All functions are stateless, i.e. one request has one response.
- Using the HTTP-API no data can be retrieved. New data can only be sent.

The following examples show you the possibilities and limitations of the HTTP API: The HTTP-API is suitable for:

- Integration of subscribe/unsubscribe forms on your website to Episerver Campaign.
- Adding data to a recipient profile by email.

The HTTP-API is inappropriate or only suitable with certain restrictions for:

- Profile change forms that directly access the Episerver Campaign database.
- Recommendation forms that directly access the Episerver Campaign database.

Restrictions

Existing data stored in the Episerver Campaign database cannot be retrieved using the HTTP-API. However, to change a recipients profile, it is absolutely necessary to retrieve data. To implement this function using the HTTP-API, the recipient data must be kept in your own database. Changes are made in this database. Before a mailing is sent, the data must be copied from your database to the Episerver Campaign database. This can be done with the HTTP-API.

For the latter functionalities, we recommend to use the SOAP-API. This convenient interface offers almost unlimited possibilities to connect Episerver Campaign to your website.

Information regarding general use

If you wish to restrict the requests of the HTTP-API to the services and operations needed for your purposes, contact customer support and tell us which services and operations you want to use.

For every HTTP-API request from the form service, Episerver sets up an IP address or IP address range from which the interface can be accessed. This setting reduces the risk of unauthorised access through the HTTP-API. Tell us the IP address or IP address range you want to allow.

You can as well restrict the access to the API server by using a firewall. If you wish to do so, we recommend to use DNS-based filter rules or the subnet 193.169.180.0/23.

Before starting the implementation of the HTTP-API on your system, read the following carefully: The correct functionality of the HTTP-API interface can only be guaranteed if the automatically sent exceptions and return values are continuously analysed by your IT department and necessary adjustments resulting from these are carried out on your system and/or website. Give us a contact person we can inform about extensions, updates and troubleshooting. The following data are required:

- First name
- Last name
- Email address
- Telephone number

Security

SSL encryption

We recommend encrypting all data transmitted. Customer support gives you more information on configuring an SSL encryption. You can access the API server using the URL https://api.broadmail.de.

Upon request, we configure your client to accept SSL requests only.

Authorization code

The URL for each request contains an authorization code which identifies either the client and the recipient list or the recipient and the sent email. If this code is incorrect, the operation is not executed.

If used in an email, the authorization code is created automatically. If you use a form on your website, the URL contains the code. To get this code, login to Episerver Campaign, open the start menu and in the Administration menu, click **API Overview** and open the **Recipient list** details – the **Authorization** code column contains the authorization code.

The authorization codes of the form service must be treated equally sensitive as a combination of user name and password. Never pass the authorization code to third parties and never use HTTP API calls of the form service directly on your websites or in mailings. HTTP API calls of the form service must always be executed by the server without exposing the used source code to others. To use HTTP operations in mailings, use the mail service of the HTTP API (refer to the section General information).

If you want to deactivate an existing authorization code, perform the following steps:

1. Open the start menu and in the **Administration** menu, click **API overview**. The API overview window opens.
2. Switch to the Recipient lists tab.
3. In the **Recipient list** list, click the recipient list that contains the authorization code.
4. Click **Manage authorization codes**. The Manage Authorization Codes window opens and displays all authorization codes of this recipient list.
5. In the **Authorization code** list, click the authorization code you want to deactivate and then click **Deactivate authorization code**.

Additionally, we recommend to use the IP restriction feature of Episerver Campaign for HTTP API calls of the form service in your client. To use this feature, contact our customer support.

IP security

We recommend to use an IP access restriction for the HTTP API form service. To setup the IP access restriction, send the IP addresses of your web server to customer support. Access from other IP addresses (than the defined ones) is denied.

Request types

Usually, the HTTP API accepts HTTP GET and POST requests. If you are sure that you use only one type of transmission, we can configure your client to accept only the desired operation.

HTTP Basics

URL structure

A URL for a HTTP-API request has the following structure (example):

```
https://api.broadmail.de/http/form/DHS45MDL2/unsubscribe?bmRecipientId=foo@bar.com
```

or:

```
https://api.broadmail.de/http/mail/SDGF-GH2-123S/updatefields?foo=bar&foo2[add]=1
```

The components of the URL have the following meaning: The root address of the HTTP-API for Episerver Campaign; called via an SSL connection.

```
https://api.broadmail.de/http/...
```

Without SSL encryption, the root address starts with:

```
http://...
```

The service selector sets the service to be used. Currently, the form service (form) or mail service (mail) are available.

```
.../form/...
```

The authorization code. The operation is performed only if the authorization code is correct:

```
.../DHS45MDL2/...
```

The operation selector defines the operation to be performed:

```
.../unsubscribe...
```

Parameters that are needed for the operation. These parameters are described in the section Parameters.

```
...bmRecipientId=foo@example.com...
```

Parameters

System parameters

Almost any operation requires several parameters. Parameters, that control the operation are prefixed with a **bm** (for example, bmRecipientId=smith@example.com).

The following system parameters are accepted for each operation:

Parameter	Description
bmSuccessUrl	If a URL is passed using this parameter, the result (success message) is forwarded to this URL after the operation was successfully performed. The GET parameter **bmResult** is appended to this URL. This parameter contains the success message, such as ok or ok:update. The **bmSuccessUrl** takes priority over the **bmUrl**.
bmFailureUrl	If a URL is passed using this parameter, the result (error message) is forwarded to this URL after the operation failed. The GET parameter **bmResult** is appended to this URL. This parameter contains the error message, such as duplicate or blacklisted. The **bmFailureUrl** takes priority over the **bmUrl**.
bmUrl	If a URL is passed using this parameter, the result is forwarded to this URL after performing the operation (no matter if successfully or failed). The GET parameter **bmResult** is appended to this URL. This parameter contains the notification message, such as ok or duplicate. This parameter can be replaced by **bmSuccessUrl** or **bmFailureUrl** to handle success and failure messages separately. **bmSuccessUrl** and **bmFailureUrl** take priority over **bmUrl** if used in the same operation.
bmEncoding	This parameter defines the character set for the transmitted data. By default, it is set to ISO-8859-1. If the characters are erroneous, try to set the value to ISO-8859-1. The encoding is particularly important if arrays of data are transmitted, e.g. when using the operations subscribe and updatefields.
bmVerbose	To receive detailed success and failure messages, set the value of this parameter to true (for example, instead of ok, the message {ok: removed is returned). By default, the value of this parameter is set to false.

Further system parameters are described in the respective operation.

Data parameters

Data parameters designate a field of the same name in the recipient list, e.g. firstname=John (the parameter **firstname** contains the value John). If the field does not exist, the parameter is ignored. Otherwise, it depends on the particular operation, what happens to the value of this parameter. Typically, the content of the field in the database is overwritten with the given value.

Data parameters can:

- overwrite the existing data ("salutation=Mr.")
- be added to the existing data ("add")
- be subtracted from the existing data ("sub")
- be prefixed to the existing data ("pre")
- be appended to the existing data ("post")

The desired operation is added to the parameter using square brackets.
Examples:

- firstname=John: Set value of the field "firstname" to John
- vote[add]=1: Increase the number of "vote" by 1
- language[post]=en: Append the value en- to the existing value of the field "language" (e. g. de-) . Then the new value would be: de-en-

Make sure to transmit only URL-encoded data parameters.

The following symbols are used for URL encoding:

- A question mark (?) marks the beginning of the part of the URL which contains the data request.
- An equals sign (=) is inserted between the name of a parameter and its value.
- An ampersand (&) separates two "parameter=value" elements.

The correct spelling of the recipient list fields of a recipient list can be found in the Recipient list details. To do this, perform the following steps:

1. Open the start menu and in the **Recipients** menu, click **Recipient lists**. The Recipient Lists window opens.
2. Click the respective recipient list.
3. Below the recipient list overview, click **Details**. The Show Recipient List Details window opens. Here you can easily check the spelling of the recipient list fields.

In a HTTP API request, you can use the display name as well as the internal name of a recipient list field.

Formatting rules

The following formatting rules apply:

Value	Format	Example
int/long	Decimal notation without any additional symbols.	`10000 or 5`
float/double	Decimal notation separated by a period (.).	`12345.67`
date/datetime	The following date formats are supported: 1. yyyyMMddHHmmss 2. yyyy-MM-dd'T'HH:mm:ssZ 3. yyyy-MM-dd HH:mm:ss 4. yyyy-MM-dd HH:mm 5. yyyy-MM-dd	2001-08-21T18:52:05+02:00 (+02:00 means CEST)

Return values

Usually, every operation returns a return value that contains a success or error message, such as OK: 32 or duplicate. You can use these return values to display a respective message in your form.

Many operations also support forwarding. After an operation has been performed, the user is forwarded to a given URL. The return message is added to that URL as an additional parameter (**bmResult**). For detailed information see the respective description of the operation.

If you use the parameter **bmVerbose=true**, a detailed message is returned (e.g. OK: already_unsubscribed).

Services and operations

Method	Description
blacklist and unblacklist	These operations generate or remove a blacklist entry.
copy and move	These operations copy or move a recipient from one recipient list to another.
createtrackingoptout and deletetrackingoptout	These operations allow you to deactivate and reactivate the use of action-based data for a recipient.
getsendstatus	This operation queries sending details for emails that have been sent using one of the two operations **sendeventmail** or **sendtransactionmail**.
nop	This operation has no function.
onlineversion	This operation shows a personalised online version of a mailing.
remove	This operation removes a recipient from a recipient list.
sendeventmail	This operation sends an event mailing to a single recipient.
sendtransactionmail	This operation creates a new recipient and sends an event mailing to this recipient.
subscribe	This operation adds a new recipient to a recipient list or updates an existing recipient.
unsubscribe	This operation unsubscribes a recipient from a recipient list.
updatefields	This operation updates recipient data fields.
uploadpersonalizedattachments	This operation uploads one ore more attachments via HTTP POST request.

Form service

The form service is used for the integration of forms (e.g. a registration form). The form service is linked by an authorization code to a specific recipient list. All operations within this service relate to this recipient list. Since the form service is usually used by a single server, we recommend to always use IP security. To select the form service, enter the service selector value form, for example:

```
http://api.broadmail.de/http/form/....
```

Mail service

The mail service is used is used for the integration of profile updates by email. You can add personal data to the recipients' profiles or move them to another list. Since the mail service is used by multiple users, IP security is not possible. The authorization code generated by the system is protected by a checksum against tampering. To select the mail service, enter the service selector value **mail**. Instead of an authorization code, insert the placeholder {bmMailId}, which is automatically replaced in each email by a value. For example:

```
http://api.broadmail.de/http/mail/{bmMailId}/unsubscribe
```

Operations

The following chapters describe the operations in more detail. To select on of these operations, enter the name of the operation.

Not all operations are available in both services (mail and form).

blacklist and unblacklist

These operations generate or remove a blacklist entry.

Blacklist entries can be individual recipients (i.e. an email address) or wildcards which block all recipients that match the defined pattern (e.g. a whole domain). Use a question mark ? for a single character and * for several characters.

These operations are available in the form service.

Parameters

Name	Mandatory	Default value	Description
bmPattern	yes	–	If the blacklist entry refers to a single recipient, the email address can be set here. To blacklist multiple recipients you can use wildcards as described above.
bmReason	no	`HTTP-API`	The reason for blacklisting.

Return values

Value	Description
`ok`	**blacklist**: Entry successfully added to the blacklist. **unblacklist**: Entry successfully removed from blacklist.
`ok: already_blacklisted`	Only with **bmVerbose=true**: The entry already exists in the blacklist.
`wrong_tag`	Authorization failed. Error pres: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`missing_id`	No **bmRecipientId** was transmitted (mandatory) when using the form service.
`system_error`	A general error occurred.

Example 1

```
.../form/.../blacklist?bmPattern=foo%40example.com
```

The recipient with the ID smith@example.com is blacklisted.

Example 2

```
.../form/.../unblacklist?bmPattern=foo%40example.com
```

The recipient with the ID smith@example.com is removed from the blacklist.

Example 3

```
.../form/.../blacklist?bmPattern=%2A%40example.com
```

All recipients with an email address containing the domain example.com are blacklisted.

copy and move

These operations copy or move a recipient from one recipient list to another.

Both operations use the same parameters. The operation copy creates a duplicate of an entry (i.e. the recipient), move creates a duplicate of an entry and then removes the original entry.

The operations copy and move are available in the mail and the form service. If you are using the mail service, we recommend to use the system parameter **bmUrl** to avoid that a white page with a status message is displayed to the recipient.

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	yes/no	–	The ID of the recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.
bmListId	yes	–	The ID of the recipient list to which the recipient is copied or moved.

Return values

Value	Description
`ok`	Entry successfully moved/copied
`ok: already_ exist`	Only with **bmVerbose=true**: Recipient already exists in the target list
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`wrong_id`	The transmitted **bmListId** is incorrect.

Value	Description
`not_found`	Only when using the form service: The **bmRecipientId** was not found neither in the source recipient list nor in the target recipient list.
`missing_id`	**bmRecipientId** and/or **bmListId** was/were not transmitted.
`system_error`	A general error occurred.

Example 1

```
.../mail/.../copy?bmListId=12345&bmUrl=http://www.google.de
```

The recipient of the email is copied from the current recipient list to the recipient list with the ID **12345**. Then, the recipient is forwarded to Google.

Example 2

```
.../form/.../move?bmListId=12345&bmRecipientId=foo%40example.com
```

The recipient with the ID smith@example.com is moved to the recipient list with the ID **12345** and removed from the current recipient list.

createtrackingoptout and deletetrackingoptout

These procedures allow you to deactivate and reactivate the use of action-based data for a recipient. These procedures are available in the form and mail service. On forms or in emails you can give your recipients the option of rejecting the use of personally identifiable data.

This recipient will then not be included in action-based target groups (e.g. **has clicked Link B in Mailing A**), even if they meet the conditions of the target group.

Notes on tracking opt-out

- Opens and clicks are still saved anonymously and included in global reports. The open and clicks in a mailing of such a recipient would therefore be counted. However, the recipient is not taken into account if you create a target group containing all recipients of this mailing who opened or clicked it.
- All personal action-based data are anonymised from the time of activation. Action data in the recipient history is not retrospectively anonymized.

Parameter

Name	Mandatory	Default value	Description
bmRecipientId	Yes	–	The recipient ID (usually the email address) This parameter is not required when using the Mail-Service. This parameter is required when using the Form service.

Return values

Value	Description
`ok`	- **createtrackingoptout**: The use of action-based data has been deactivated for the recipient. - **deletetrackingoptout**: The use of action-based data has been reactivated for the recipient.
`vetoed`	The execution of this operation was prevented because it is deactivated. Ask our customer support to activate this feature.
`wrong_tag`	Authorization failed. Error codes: - 501=wrong authentication tag - 502=wrong request IP

Value	Description
	» 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`missing_id`	No **bmRecipientId** was transmitted (mandatory) when using the form service.
`system_error`	A general error has occurred.

Example 1

```
.../form/.../createtrackingoptout?bmRecipientId=john.deer@example.com
```

The use of action-based data has been deactivated for the recipient with ID john.deer@example.com.

Example 2

```
.../form/.../deletetrackingoptout?bmRecipientId=john.deer@example.com
```

The use of action-based data has been reactivated for the recipient with ID john.deer@example.com.

getsendstatus

This operation queries sending details for emails that have been sent using one of the two operations sendeventmail or sendtransactionmail.

This operation is available in the form service.

Parameters

Name	Mandatory	Default value	Description
bmMailId	yes	–	The ID of the email whose sending status you want to check.

Return values

Value	Description
`enqueued`	The email is still in the queue.
`sending`	The sending process has not been finished yet.
`sent`	The email has been sent successfully.
`could_not_be_sent`	An error occurred during the sending process.
`missing_id`	No **bmMailId** was transmitted.
`vetoed`	The sending of the email was blocked. This may have several reasons, for example, the number of coupon codes left may be too low.
`wrong_mailing_status`	The mailing with the given bmMailingId has been stopped or is finished.
`blacklisted`	The recipient is blacklisted and will not receive any further emails.
`too_many_bounces`	The recipient has exceeded the bounce limit and will not receive any further emails.
`filtered`	Due to target groups used by the mailing, the recipient did not receive an email.
`not_found`	The recipient with the given **bmRecipientId** could not be found in the recipient list.
`unsubscribed`	The recipient has unsubscribed from the service and will not receive any further

Value	Description
	emails.
`system_error`	A general error occurred.

The return values of this operation are available for a maximum of 14 days after sending a mailing.

Example 1

```
.../form/.../getsendstatus?bmMailId=4LE98TN-C01-H3SD90
```

The transmitted **bmMailId** must be the return value of the respective sendeventmail or sendtransactionmail call.

nop

This operation has no function.

It is used to test authorization or forwarding to a URL.

The operation **nop** is available in the mail and the form service.

Parameters

Name	Mandatory	Default value	Description
any parameter	no	–	For testing purposes only.

Return values

Value	Description
`ok`	Operation **nop** successfully performed; i.e. the test was successful.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`system_error`	A general error occurred.

Example 1

```
.../mail/.../nop? bmUrl=http://www.google.de
```

The recipient of the email is forwarded to Google if she/he clicks any link.

onlineversion

This operation shows a personalized online version of a mailing.

You can select a specific mailing by its internal name or search your mailings using the **pattern** parameter. If more than one mailing matches the pattern, the latest one is selected. Also, if you only specify the recipient ID, the latest mailing is displayed. Use this operation, for example, to create a link to the latest newsletter. This link can be used in confirmation emails or on your website. Unlike other operations, this operation always performs forwarding to a web page, where the online version of the mailing is shown. Thus, no bmUrl or bmSuccessUrl parameters have to be transmitted.

This operation is available in the mail and the form service.

Warning: Never use this HTTP-API request directly on a web page or in a mailing. HTTP API requests of the form service must always be executed by the server without exposing the used source code to others. To use HTTP operations in mailings, use the mail service of the HTTP API (refer to the section General information).

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	no	–	The ID of the new recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.
bmPattern	no	–	With the pattern parameter you can search for mailings, which match this pattern. The pattern refers to the internal names of the mailings. You can use wildcards to define a pattern. Use a question mark ? for a single character and * for several characters. If more than one mailing is found, the latest one will be selected.
bmMailingId	no	–	This parameter explicitly transmits the ID of the mailing you want to display. If **bmPattern** and **bmMailingId** are transmitted, the **bmMailingId** has priority.

Return values

Value	Description
`ok`	Recipient forwarded to the online version of the email
`missing_id`	The recipient with the given **bmRecipientId** does not exist in the recipient list or no appropriate mailing was found.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`system_error`	A general error occurred.

Example 1

```
.../mail/.../onlineversion
```

The recipient of the email is forwarded to the last mailing sent by the same client.

Example 2

```
.../form/.../onlineversion?bmRecipientId=foo%40example.com&bmPattern=sonder*
```

The recipient of the email is forwarded to the last mailing sent by the same client, whose name starts with **special**.

remove

This operation removes a recipient from a recipient list.

This operation does not unsubscribe a recipient. To unsubscribe a recipient, use the **unsubscribe** operation.

This operation is available in the mail and the form service.

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	yes/no	–	The ID of the new recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.

Return values

Value	Description
`ok`	Recipient successfully removed from the recipient list
`ok: not_found`	Only with **bmVerbose=true**: The recipient could not be found in the recipient list.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`missing_id`	When using the form service: No **bmRecipientId** was transmitted.
`system_error`	A general error occurred.

Example 1

```
.../form/.../remove?bmRecipientId=foo%40example.com
```

The recipient with the ID **smith@example.com** is removed from the recipient list. This recipient list is identified by the authorization code.

Example 2

```
.../mail/.../remove? bmUrl=http://www.google.de
```

The recipient of the email is removed from the recipient list the mailing was sent to. Then, the recipient is forwarded to Google.

sendeventmail

This operation sends an event mailing to a single recipient.

Use this function to send, for example, confirmation emails, notifications or status updates to existing recipients. In some cases, it may be useful to use the sendtransactionmail operation instead.

This operation is available in the mail and the form service.

Warning: **Never use this HTTP API request directly on a web page or in a mailing**. HTTP API requests of the form service must always be executed by the server without exposing the used source code to others. Otherwise there is a potential risk that other web user read these data and send emails from your client and at your expense. To use HTTP operations in mailings, use the mail service of the HTTP API (refer to the section General information).

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	yes/no	–	The ID of the recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.
bmMailingId	yes	–	The ID of the event mailing to be sent.

Return values

Value	Description
`enqueued: <bmMailId>`	The sending process has been started. Use the operation getsendstatus to get the detailed sending status.
not_found	The mailing with the given **bmMailingId** could not be found.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol

Value	Description
	» 505=wrong recipient list » 506=wrong action » 507=action not found
`missing_id`	No **bmMailingId** was transmitted.When using the form service: No **bmMailingId** and/or no **bmRecipientId** was/were transmitted.
`syntax_error`	The transmitted **bmRecipientId** is invalid.
`wrong_mail-ing_type`	The mailing with the given **bmMailingId** is not of the type **event**.
`wrong_mail-ing_status`	The mailing with the given **bmMailingId** has been stopped or is finished.
`system_error`	A general error occurred.

Example

```
.../form/.../sendeventmail?bmRecipientId=foo%40example.com&bmMailingId=12345
```

The mailing with the ID **12345** is sent to the recipient with the ID **smith@example.com**.

sendtransactionmail

This operation creates a new recipient and sends an event mailing to this recipient.

Additional parameters for personalization can be transmitted. You can also send fax, SMS or printed messages (see Sending print messages), if your client is configured accordingly. This operation can also be used to send an SMS or a fax if your client is configured accordingly. In certain cases this operation can also be used to send event mailings several times to already existing recipients.

Notes on the usage of this operation: You can use this operation to send emails to existing recipients. Each call of this operation creates a new recipient dataset. Unsubscribes and bounces are always associated with the ID field (which may be, for example, the email address) of the recipient list used. Use this operation only for confirmation emails, notifications, status emails or new subscriptions in case recipients forgot their password. Note that some of these transaction emails must not contain promotional content.

This operation is available in the form service.

Warning: Never use this HTTP-API request directly on a web page. HTTP API requests of the form service must always be executed by the server without exposing the used source code to others. Otherwise there is a potential risk that other web user read these data and send emails from your client and at your expense.

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	yes	–	If the mailing is of the type email, the parameter must contain the email address of the recipient. If an SMS or fax is sent, the parameter must contain the telephone number of the recipient.
bmMailingId	yes	–	The ID of the mailing (type: event) to be sent. The mailing must be associated with the same recipient list used by the authorization code.

Name	Mandatory	Default value	Description
bmPersonalizedAttachmentsToken	no	–	To send an attachment, set the value of this parameter to the return value of the operation uploadpersonalizedattachments (i.e. the token). The personalized attachment feature must first be activated by customer support.
bmSanitize	no	–	Set the value of this parameter to true to convert potentially dangerous characters in the supplied recipient data into their respective HTML equivalents .
...	no	–	Any parameter that corresponds to a parameter of the recipient list.

Return values

Value	Description
`enqueued: <bmMailId>`	The sending process has been started. Invoke the operation getsendstatus to get sending status details.
`not_found`	The transmitted mailing with the given **bmMailingId** could not be found.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`missing_id`	No **bmMailingId** and/or no **bmRecipientId** was/were transmitted.

Value	Description
`syntax_error`	The transmitted **bmRecipientId** or another parameter is invalid.
`syntax_error: Invalid bmPer-sonalizedAttachmentsToken.`	The token is either expired, was not created for this client or has been tampered with.
`wrong_mailing_type`	The mailing is not of the type **event**.
`wrong_mailing_type: Personalized attachments can only be used with EMAIL-mailings.`	Personalized attachments can only be used for mailings of the media type **email**.
`wrong_mailing_status`	The mailing with the given **bmMailingId** has been stopped or is finished.
`system_error`	A general error occurred.

Example 1

```
.../form/.../sendtransactionmail?bmRecipientId=foo%40example.com&bmMailingId=12345
```

The mailing with the ID **12345** is sent to the recipient with the ID **smith@example.com**.

Example 2

```
.../form/.../sendtransactionmail?bmRecipientId=foo%40example.com&
bmMailingId=12345&bmPersonalizedAttachmentsToken=1a2b3c456
```

The mailing with the ID **12345** is sent to the recipient with the ID **smith@example.com** along with a personalized attachment (using the token **1a2b3c456**). For more examples for using the personalized attachments, refer to section Personalized Attachments—Examples.

Sending print messages

To also send print messages with this operation, use the operation as described, do not enter a **bmRecipientId** and then apply the following additional print recipient list fields:

Recipient list field name	Mandatory	Notes
bmletteraddress_ salutation	No	Contains the salutation
bmletteraddress_ firstname	No	Contains the first name

Recipient list field name	Mandatory	Notes
bmletteraddress_lastname	Yes	Contains the last name If you wish to use only the company name for business addresses (without providing the name of a member of staff), this must be entered here.
bmletteraddress_extraaddressline	No	Contains the extra address line  If this is a business address and you have entered the name of a staff member in the **last name** field, then enter the company name into this field. The letter cannot be delivered without this.
bmletteraddress_street	Yes	Contains the street name Mandatory field along with **Number**.  The **Street and Number** and **PO box** fields must be left empty.
bmletteraddress_streetnumber	Yes	Contains the building number Mandatory field along with **Street**.  The **Street and Number** and **PO box** fields must be left empty.
bmletteraddress_streetandnumber	No	Contains the street and building number  When using this field, the **Street**, **Number** and **PO box** fields must be left empty.
bmletteraddress_postofficebox	No	Contains the PO Box

Recipient list field name	Mandatory	Notes
		When using this field, the **Street**, **Number** and **Street and Number** fields must be left empty.
bmletteraddress_zipcode	Yes	Contains the postcode
bmletteraddress_city	Yes	Contains the town/city
bmletteraddress_countrycode	Yes	Contains the country code. This field must be set to **DE**. Dispatch is only available in within Germany.

subscribe

This operation adds a new recipient to a recipient list or updates an existing recipient.

The operation **subscribe** is available in the mail and the form service. To update recipient data using the mail service, we recommend to use the operation updatefields instead.

Stick to the formatting rules when transmitting recipient data.

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	Yes/No	–	The ID of the new recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.
bmOptInId	No	–	The ID of an opt-in process. The opt-in process will be started, when the parameter is transmitted. If there is a pending opt-in process the recipient has not finished yet, a new opt-in process is started and no error message is returned.
bmOptinFrequencyLimit	No	-	Use this parameter to set a period (in minutes), that defines how long the last opt-in process of the recipient must date back so that a new opt-in process can be started. If, for example, a new recipient is created and a value of 60 is given, then the last opt-in process must date back at least 60 minutes. Otherwise no new opt-in process is started and no further confirmation email (opt-in email) will be sent to this recipient.  This parameter is only relevant with a double opt-in process.
bmFailOnUnsubscribe	No	`false`	By default, unsubscribed recipients will be re-

Name	Mandatory	Default value	Description
			activated when subscribed again. If this parameter is set to true, unsubscribed recipients are skipped and a message is returned (unsubscribed).
bmOverwrite	No	`false`	By default, recipients who already exist are recognised as duplicates and a duplicate message is returned. When you set the parameter to true, all existing data are overwritten with the new data. No **duplicate** message is returned.
bmOptinSource	No	–	Optional parameter that defines the opt-in source of the recipient. The parameter can only be set using this operation. It is not possible to set or change the value of this parameter afterwards using the broadmail user interface.
bmSanitize	no	–	Set the value of this parameter to true to convert potentially dangerous characters in the supplied recipient data into their respective HTML equivalents .
...	No	–	Any parameter that corresponds to a parameter of the recipient list.

Return values

Value	Description
`ok`	Recipient successfully added to the recipient list
`ok: updated`	Only with **bmVerbose=true**: Recipient data successfully updated
`optin_already_started`	Only with double opt-in and a given value for **bmOptinFrequencyLimit**: Another subscribe operation was triggered for the same recipient, but no opt-in process was triggered, because the new subscription was sent during the pre-defined blocking period.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method

Value	Description
	» 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`duplicate`	Dataset already exists (no overwriting).
`unsubscribed`	Recipient is unsubscribed.
`syntax_error`	Syntax error in the parameter, e.g. the email address is invalid.
`too_many_bounces`	The recipient is or has already been subscribed and has exceeded the bounce limit.
`blacklisted`	Recipient is blacklisted.
`system_error`	A general error occurred.

Example 1

```
.../subscribe?bmRecipientId=foo%40example.com&bmOptInId=2131232
```

Adds the recipient with the ID **foo@example.com** to the recipient list and starts opt-in process **2131232** for this recipient.

If the recipient already exists, the return value **duplicate** is sent. No opt-in process is started.

Example 2

```
.../subscribe?bmRecipientId=foo%40example.com&bmOverwrite=true&bmOptInId=2131232
```

Adds the recipient with the ID **foo@example.com** to the recipient list and starts opt-in process **2131232** for this recipient.

If this recipient is already available, the existing data are overwritten and updated with the new data. A new opt-in process is not started.

If the recipient is on the unsubscribe list, he will be removed from this and overwritten in the recipient list. No opt-in process is started.

Example 3

```
.../subscribe?bmRecipientId=foo%40example.com&bmOverwrite=false&bmOptInId=2131232
```

Adds the recipient with the ID **foo@example.com** to the recipient list and starts opt-in process **2131232** for this recipient.

If the recipient already exists, the return value **duplicate** is sent. The existing data are not overwritten. A new opt-in process is not started.

unsubscribe

This operation unsubscribes a recipient from a recipient list. By default, unsubscribed recipients are not deleted from the list, but marked as **unsubscribed**.

This operation is available in the mail and the form service.

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	Yes/No	–	The ID of the recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.
bmMailId	no	–	This ID identifies the recipient to be unsubscribed and the received email. If not transmitted in the form service, the unsubscription is valid, but cannot be assigned to a specific mailing.
bmRemoveId	no	`false`	If set to true, the recipient is removed from all recipient lists. Recipients contained in test lists and transaction mailing lists cannot be deleted with this parameter.

Return values

Value	Description
`ok`	Recipient successfully unsubscribed from recipient list
`ok: not_found`	Only with **bmVerbose=true** and **bmRemoveId=true**: The recipient to be removed was not found in any recipient list.
`ok: already_unsubscribed`	Only with **bmVerbose=true**: Recipient is already unsubscribed. In this case, the recipient cannot be removed from the recipient list.
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol

Value	Description
	» 505=wrong recipient list » 506=wrong action » 507=action not found
`missing_id`	No **bmRecipientId** was set (mandatory) when using the form service.
`system_error`	A general error occurred.

Example 1

```
.../form/.../unsubscribe?bmRecipientId=foo%40example.com
```

Unsubscribes the recipient with the ID smith@example.com using the form service.

Example 2

```
.../mail/.../unsubscribe?bmRemoveId=true&bmUrl=http://www.google.de
```

Unsubscribes the recipient of the email and removes the recipient from all recipient lists. Then, the recipient is forwarded to Google.

updatefields

This operation updates recipient data fields. Field values can be overwritten, added, subtracted, prepended or appended.

This operation is available in the mail and the form service. If you are using the mail service, we recommend to use the system parameter bmUrl to avoid that a white page with a status message is displayed to the recipient.

Transactional recipient lists (recipient lists that can only be used for API calls) cannot be updated using this operation.

Parameters

Name	Mandatory	Default value	Description
bmRecipientId	yes/no	–	The ID of the recipient (usually the email address). If used in the mail service, this parameter is not mandatory (the email address is sent anyway). If used in the form service, the parameter is mandatory.
bmNewRecipientId	no	–	If you want to change the **bmRecipientId** of a recipient, use this parameter, since the parameter **bmRecipientId** is not a regular data field, but an identifier.  Changing the ID of a recipient also affects all other recipient lists of the same client.
bmOverwrite	no	`false`	If you want to change the **bmRecipientId** of a recipient (see previous parameter) and the specified **bmRecipientId** already exists, an error message is returned and the operation is cancelled. If you set the value of **bmOverwrite** to true, the existing value is overwritten and no error message is returned.
bmSanitize	no	–	Set the value of this parameter to true to convert potentially dangerous characters in the supplied recipient data into their respective HTML equivalents .

Name	Mandatory	Default value	Description
...	no	–	Any parameter that corresponds to a parameter of the recipient list.

Return values

Value	Description
`ok`	Recipient data successfully updated
`wrong_tag`	Authorization failed. Error codes: » 501=wrong authentication tag » 502=wrong request IP » 503=wrong request method » 504=wrong protocol » 505=wrong recipient list » 506=wrong action » 507=action not found
`already_exist`	Only with **bmOverwrite=false**: The new recipient ID transmitted using the parameter **bmNewRecipientId** already exists. To avoid this error and overwrite the existing ID, set **bmOverwrite** to true.
`missing_id`	No **bmRecipientId** was transmitted (mandatory) when using the form service.
`not_found`	No entry with the given **bmRecipientId** found
`system_error`	A general error occurred.

Example 1

```
.../mail/.../updatefields?interesse=ja&bmUrl=http://www.google.de
```

The dataset field **interests** of the recipient of the email is set to yes. Then, the recipient is forwarded to Google.

Example 2

```
.../mail/.../updatefields?myclicks[add]=3&hasclicked=true&bmUrl=http://www.google.de
```

The value of the dataset field **myclicks** of the recipient of the email is increased by 3. The dataset field **hasclicked** is set to true. Then, the recipient is forwarded to Google.

Example 3

```
.../form/.../updatefields?myclicks[add]-
]=3&hasclicked=true&bmRecipientId=foo%40example.com
```

The value of the dataset field **myclicks** of the recipient with the ID **smith@example.co**m is increased by 3. The dataset field **hasclicked** is set to true.

Example 4

```
.../-
form/.../up-
datefields?bmRecipientId=f1%40example.com&bmNewRecipientId=f2%40example.com
```

The ID of the recipient **f1@example.com** is changed to **f2@example.com**.

uploadpersonalizedattachments

This operation uploads one ore more attachments via HTTP-POST request. These attachments can be used when sending transaction emails.

For more information regarding the personalized attachments feature, refer to the section Personalized Attachments—Examples.

This operation is available in the form service.

Hints

- The Personalized Attachment feature must first be activated by the customer support.
- Personalized attachments can only be used for event mailings.
- You must not upload more than 500 kB at once.
- With each call of this operation up to five files can be uploaded.
- An attachment is valid for 24 hours by default; i.e. after this time the attachment cannot be used for sending anymore.

Parameters

Name	Mandatory	Default value	Description
bmFile	yes	–	Defines the attachment to be uploaded. This parameter may occur multiple times. I.e. multiple attachments (up to five) can be uploaded per call.

Return values

Value	Description
`ok: <token>`	The attachment has been successfully uploaded and can be used later using this returned **token** with the operation sendtransactionmail. A token is valid for 24 hours by default; i.e. after this time the token becomes invalid and you must invoke the operation again.
`syntax_error: No files uploaded.`	The value of the parameter **bmfile** is invalid. Check the spelling.
`syntax_error: Invalid file-parameter name '<Para-`	Instead of the parameter name **bmFile** an incorrect parameter name was used. Always use the parameter name **bmfile** to upload attachments.

Value	Description
`meter name>' - must be 'bmFile'.`	
`vetoed: Personalized attachments are not enabled.`	The personalized attachments feature has not yet been activated. To activate the feature, contact customer support.
`file_upload_failed: Ask broadmail support for ticket #<ticket ID>`	A general error occurred while uploading the attachment. Contact customer support and quote this ticket ID.

Example

The examples for personalized attachments can be found in the section Personalized Attachments—Examples.

HTTP Examples

Example	Description
Integration into your website	Integration examples for your website
Personalized Attachments—Examples	Examples for the Personalized Attachments feature

Integration into your website

Simple registration form

This is a typical example for the form service. A simple registration form on your website allows users to register for your newsletter. The user enters personal data and submits it to the database. The example uses PHP implementation and consists of two pages: The actual registration page and a Thank you page which opens, when the registration is done.

The static site **form.php** generates the form with the input fields and leads to a dynamic site **thank_you.php** which transmits the data to the HTTP-API.

Form page form.php

```
<html>
<head>
<title>Newsletter registration</title>
</head>
<body>
<form action="thank_you.php" method="post">
  <p>
    <b>Salutation:</b>
    <select name="salutation" id="salutationid">
       <option value="Mr.">Mr.</option>
       <option value="Mrs.">Mrs.</option>
     </select><br>
    <b>First name:*</b>
    <input type="text" name="firstname"><br>
    <b>Surname:*</b>
    <input type="text" name="lastname"><br>
    <b>Email:*</b>
    <input type="text" name="email"><br>
    <input type="submit" name="send" value="send" class="submit" />
  </p>
</form>
```

```
</body>
</html>
```

Page thank_you.php

```
<?php
$url  = 'https://api.broadmail.de/http/form/FOO-BAR-S33D/subscribe';
$url .= '?bmRecipientId=' . urlencode(utf8_decode($_POST['email']));
$url .= '&salutation=' . urlencode(utf8_decode($_POST['salutation']));
$url .= '&firstname=' . urlencode(utf8_decode($_POST['firstname']));
$url .= '&lastname=' . urlencode(utf8_decode($_POST['lastname']));
$result = @file_get_contents($url);
?>
<html>
<head>
<title>Newsletter Registration</title>
</head>
<body>
<p>
<?php
  if ($result == 'ok') {
?>
      <h1>Thank you for your registration.</h1>
<?php
  } else if ($result == 'duplicate') {
?>
      <h1>You are already registered.</h1>
<?php
  } else {
?>
      <h1>An error occurred, please try again.</h1>
<?php
  }
?>
<p>
</body>
</html>
```

Personalized Attachments—Examples

The Personalized Attachment feature must first be activated by customer support.

The Personalized Attachments enables you to send individual attachments to particular recipients. Before sending, these attachments must be transmitted to Episerver Campaign using the HTTP-API. The uploaded attachments can be used when sending transaction emails.

Usage and functionality

Using Personalized Attachments you can, for example, easily and automatically send invoices and orders to your recipients. You can transmit these files (PDF, text file or another file format) to Episerver Campaign using the uploadpersonalizedattachments operation. When calling this operation it is possible to upload one or more attachments using an HTTP request. Therefore, the content of the POST requests consists of a multi-part message of the content type **multipart/form-data**. A correct header of the request can be structured as follows:

```
connection: Keep-Alive
content-length: 1332
content-type: multipart/form-data;
boundary=rTEJvW_pp2jgbYyxtf7sa8eXTJ1qUE8eizBIT
host: api.broadmail.de
user-agent: Apache-HttpClient/4.2 (java 1.5)
```

Additionally, the attribute boundary is defined here. It is used to separate the content of the individual files in the post request from each other.

The content of the POST request to upload two files looks as follows (abstract):

```
--<Boundary>
Content-Disposition: form-data; name="bmFile"; filename="Your_Order.txt"
Content-Type: text/plain
Content-Transfer-Encoding: binary
<Text content>
--<Boundary>
Content-Disposition: form-data; name="bmFile"; filename="Your_Order.pdf"
Content-Type: application/pdf
Content-Transfer-Encoding: binary
<Content of the PDF file>
--<Boundary>--
```

The attribute **filename** defines the file name of the attachment. This file name is used when sending the mailing. This way you can transmit recipient-related file names, such as **Invoice_John_Smith.pdf**. If the attachment was successfully uploaded, a token is returned:

```
ok: a16dbd294d700428a5704aca679d9a887489cca1098cc1bd7de54bd523a475eb8437813
```

To use the uploaded attachments in transaction emails, you have to transmit the above mentioned token when calling the operation sendtransactionmail. Such a call might look as follows:

```
https://api.broadmail.de/http/form/<authorisation code>/
sendtransactionmail?bmRecipientId=<Email Address>&bmMailingId=<Mailing ID of the spe-
cial mailing>
&bmPersonalizedAttachmentsToken=<Returned token>
```

In this example, the recipient receives the content of the mailing and the attachments referenced by the token.

Further examples

Below you can find some examples for using the personalized attachment. The placeholder **<auth_code>** used in these examples, must be replaced with the actual authorization code.

Java

Im In the following example the file **Your_Order.pdf** is uploaded and send to the recipient with the email address **john.smith@example.co**m. This example uses some Apache HttpComponents. You can get the required libraries from http://hc.apache.org.

```
public static void main(String[] args) throws Exception {
    final String token = uploadPersonalizedAttachments();
    sendTransactionMail(token);
}

private static String uploadPersonalizedAttachments() throws Exception {
    final MultipartEntity uploadMultipartEntity = new MultipartEntity();
    uploadMultipartEntity.addPart("bmFile", new FileBody(new File("Your_Order-
.pdf")));
    final HttpPost httpPost = new HttpPost
("https://api.broadmail.de/http/form/<auth_code>/
uploadpersonalizedattachments");
    httpPost.setEntity(uploadMultipartEntity);
    final HttpClient uploadHttpClient = new DefaultHttpClient();
    final HttpResponse uploadResponse = uploadHttpClient.execute(httpPost);
    final HttpEntity uploadResponseEntity = uploadResponse.getEntity();
    final ByteArrayOutputStream uploadByteArrayOutputStream = new ByteAr-
rayOutputStream();
    uploadResponseEntity.writeTo(uploadByteArrayOutputStream);
    final String answer = uploadByteArrayOutputStream.toString("UTF-8");
    if(uploadResponse.getStatusLine().getStatusCode() != 200 || !answer.startsWith
("ok: ")) {
        throw new RuntimeException(answer);
    }
    return answer.replace("ok: ", "");
}
```

```
private static void sendTransactionMail(String token) throws Exception {
    final MultipartEntity sendMultipartEntity = new MultipartEntity();
    sendMultipartEntity.addPart("bmRecipientId", new StringBody("john.-
smith@example.com"));
    sendMultipartEntity.addPart("bmMailingId", new StringBody("123456"));
    sendMultipartEntity.addPart("bmPersonalizedAttachmentsToken", new StringBody
(token));
    final HttpPost sendHttpPost = new HttpPost("https://ap-
i.broadmail.de/http/form/<auth_code>/sendtransactionmail");
    sendHttpPost.setEntity(sendMultipartEntity);
    final HttpClient sendHttpClient = new DefaultHttpClient();
    final HttpResponse sendResponse = sendHttpClient.execute(sendHttpPost);
    final HttpEntity sendResponseEntity = sendResponse.getEntity();
    final ByteArrayOutputStream sendByteArrayOutputStream = new ByteAr-
rayOutputStream();
    sendResponseEntity.writeTo(sendByteArrayOutputStream);
    System.out.println(sendByteArrayOutputStream.toString("UTF-8"));
}
```

CURL

The following shell script **upload_attachment_and_send_to_recipient.sh** uploads an attachment using CURL and sends it in a transaction email:

```
#! /bin/sh
URL="https://api.broadmail.de/http/form/<auth_code>"
MAILING="123456"
FILE="${1}"
RECIPIENT="${2}"

if  [ -z "${FILE}" ] || [ -z "${RECIPIENT}" ]; then
    echo "Some argument is missing. Please use following syntax: '$0 <file> <recip-
ient>'"
    exit 1
fi

echo "Executing request to upload attachment ..."
uploadAnswer=$(curl -k -F bmFile=@${FILE} "${URL}/uploadpersonalizedattachments")
if echo "${uploadAnswer}" | grep "ok: "; then
    token=$(echo "${uploadAnswer}" | grep "ok: " | sed 's/ok: //')
    echo "Executing request to send transaction mail with token '${token}' ..."
    echo $(curl -k "${URL}/sendtrans-
actionmail?bmRecipientId=${RECIPIENT}&bmMailingId=${MAILING}
&bmPersonalizedAttachmentsToken=${token}")
else
    echo ${uploadAnswer}
```

```
    exit 1
fi
```

To invoke the above mentioned file, the following command must be invoked:

```
sh upload_attachment_and_send_to_recipient.sh "Your_Order.pdf" "john.-
smith@example.com"
```

PHP (Version 5.3.5)

In the following example the file **Your_Order.pdf** is uploaded and send to the recipient with the email address **john.smith@example.com**.

```
<?php
$file = "Your_Order.pdf";
$authTag = '<auth_code>';
$recipientId = 'john.smith@example.com';
$mailingId = '123456';
$boundary = '---------------------' . substr(md5(rand(0, 32000)), 0, 10);
$pathinfo = pathinfo($file);
$data = '--' . $boundary . "\r\n" .
    'Content-Disposition: form-data; name="bmFile"; filename="' . $pathinfo['base-
name'] . '"' . "\r\n" .
    'Content-Type: application/binary' . "\r\n" .
    'Content-Transfer-Encoding: binary' . "\r\n\r\n" .
    file_get_contents($file) . "\r\n" .
    '--' . $boundary . '--' . "\r\n";
$uploadUrl = 'https://api.broadmail.de/http/form/' . $authTag . '/up-
loadpersonalizedattachments';
$context = stream_context_create(
    array(
        'http' => array(
            'method' => 'POST',
            'header' => 'Content-Type: multipart/form-data; boundary=' . $boundary,
            'content' => $data
        )
    )
);
if (!($uploadHandle = fopen($uploadUrl, 'rb', false, $context))) {
    throw new Exception('Problem with ' . $uploadUrl);
}
$response = stream_get_contents($uploadHandle);
fclose($uploadHandle);
$token = str_replace('ok: ', '', $response);
if ($response === $token) {
    throw new Exception($response);
}
```

```
$sendUrl = 'https://api.broadmail.de/http/form/' . $authTag . '/sendtransactionmail'
.
    '?bmRecipientId=' . urlencode(utf8_decode($recipientId)) .
    '&bmMailingId=' . urlencode(utf8_decode($mailingId)) .
    '&bmPersonalizedAttachmentsToken=' . urlencode(utf8_decode($token));
$result = file_get_contents($sendUrl);
echo $result;
?>
```