Yandex.Mail for Domains API

Developer's reference

15.01.2016

Yandex

Yandex.Mail for Domains API. Developer's reference. Version
Document build date: 15.01.2016.
This volume is a part of Yandex technical documentation.
Yandex helpdesk site: http://help.yandex.ru

Copyright Disclaimer

Contact information

Yandex LLC
http://www.yandex.com
Phone: +7 495 739 7000
Email: pr@yandex-team.ru
Headquarters: 16 L'va Tolstogo St., Moscow, Russia 119021

Contents

Documentation

This section includes technical documentation for the programming interface (API) that is provided by the Yandex.Mail for Domains service.

Yandex.Mail for Domains API

This document describes the API methods that are used for managing domains. Methods are provided for managing the domain as a whole, as well as for managing the accounts of individual users. The methods can be invoked directly from programming code (for example, code on your website). A detailed description of each method is provided, including the syntax, parameters, and response format.

Sections in this document:

- Domain management – Methods that apply to the entire domain: registering and removing a domain, operations with logos, and managing the list of administrators.
- User accounts – Managing user mailboxes.
- Distribution lists – Managing distribution lists that are shared for the domain.
- Authorizing users – Managing user authentication with temporary OAuth tokens.
- Importing email – Managing mailbox migration.

DNS API

This document is a reference of Yandex.Mail for Domains API methods that are used for managing your domain's DNS.

The methods described are used for various DNS record formats: A, AAAA, CNAME, MX, NS, SRV, TXT, and SOA.

API for hosters and registrars

This document is intended specifically for hosting providers, registrars, and domain parkers. It can be used as an API reference, or as an instruction manual for connecting user domains to Yandex.Mail.

The information provided in the document will help hosting providers, registrars, and domain parkers connect their users' domains to Yandex.Mail and import their mail to Yandex, as well as get the status of the connection and mail import process.

The connection is set up automatically (in a single click) and is implemented using the API methods described in this document.

Yandex.Mail for Domains API

Describes the methods of the Yandex.Mail for Domains API.

The Yandex.Mail for Domains API lets you administer your mailboxes from programming code (for example, using the code on your web site).

The API is an HTTP(S) interface for calls. Each call performs an action. Parameters are passed in the request string or request body. All parameters should be passed using urlencode encoding. The result is output in the response body.

The table below lists the methods and their use.

Method	Use
get_token	Getting an access token
Domain management	
reg_domain	Signing up a domain
reg_default_user	Setting a default mailbox for the domain
del_domain	Disconnecting a domain
add_logo	Adding a domain logo
del_logo	Removing a domain logo
get_domain_users	Getting a list of mailboxes
check_user	Verifying the existence of a user
add_admin	Adding an additional domain administrator
del_admin	Removing an additional domain administrator
get_admins	Getting a list of additional domain administrators
User accounts	
reg_user_token	User registration
reg_user_crypto	Creating a user with an encrypted password
reg_user	Creating a mailbox in a secondary domain
get_mail_info	Getting the number of unread messages
get_user_info	Getting user data
edit_user	Changing user data
set_forward	Setting up forwarding
get_forward_list	Getting a list of forwarding and filters
delete_forward	Removing forwarding and filters
delete_user	Removing a user
del_user	Removing a mailbox in a secondary domain
Distribution lists	
create_general_maillist	Creating a general distribution list
delete_general_maillist	Removing a general distribution list
Authorizing users	
user_oauth_token	Getting a temporary access token
passport	Authorization using a temporary token
set_mail_callback	Setting an authorization callback URL
Importing email	
set_domain	Saving import settings for a domain
start_import	Starting mailbox import

Method	Use
check_import	Checking the status of the mail import process
stop_import	Stopping mail import
reg_and_imp	Registering a user and starting mail import
import_imap	Starting folder import over IMAP

Getting an access token

The `get_token` method is used for getting an access token.

The access token is used for activating the Yandex.Mail for Domains API. You only need to get the token once.

To get a token, you should have a domain that is already signed up, authenticate as the domain administrator, and go to the address shown below.

To get a token, you should have a domain that is already signed up, authenticate as the domain administrator, and go to the address shown below.

Request syntax

`domain_name` Name of the domain.

Example

Getting a token for the domain `example.com`:

Response format

If no errors occurred, the method returns `<ok token="..."/>`; in the case of errors, it returns `<error reason='...'/>`.

Domain management

Describes the Yandex.Mail for Domains API methods that are used for managing domains.

Method	Use
reg_domain	Signing up a domain
reg_default_user	Setting a default mailbox for the domain
del_domain	Disconnecting a domain
add_logo	Adding a domain logo
del_logo	Removing a domain logo
get_domain_users	Getting a list of mailboxes
check_user	Verifying the existence of a user
add_admin	Adding an additional domain administrator
del_admin	Removing an additional domain administrator
get_admins	Getting a list of additional domain administrators

reg_domain

This method is used for signing up a domain.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.
domain	Name of the domain.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>
      <secret_name>...</secret_name>
      <secret_value>...</secret_value>
    </domain>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
secret_name*	Secret name of the file or subdomain. Used for domain verification using a file or CNAME, similarly to the way this is done when connecting a domain manually.
secret_value*	Secret string (for a secret file). Used for domain verification using a file or CNAME, similarly to the way this is done when connecting a domain manually.

If the domain is already connected, the reg_domain method does not do anything when it returns the secret name and secret string.

reg_default_user

The method is used for creating a default mailbox for the domain.

The default mailbox is the mailbox that any messages are sent to that are addressed to non-existent mailboxes on this domain.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.

domain	Name of the domain.
login	Name of the mailbox. The mailbox named login should already exist.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>

      <default_email>...</default_email>

    </domain>
  </domains>
</action>
```

success*	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
default_email*	Name of the default mailbox (user login).

del_domain

This method is used for disconnecting a domain.

A disconnected domain is no longer shown in the list of domains. After disconnecting a domain, it can be reconnected again.

Note:
Disconnecting a domain does not cause any changes in the MX records. MX records must be configured separately on the DNS servers the domain is delegated to.

Request syntax

```
https://pddimp.yandex.ru/api/del_domain.xml ?
token=<access token>
 & domain=<domain name>
```

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain “manually” and getting a token for it using the get_token method.
domain	Name of the domain.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>
    </domain>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.

add_logo

This method is used for adding a domain logo.

The method is called only as a POST request. The file containing the logo, along with its parameters, are passed as multipart/form-data. Image files in JPG, GIF or PNG formats are supported, with a size up to 2 Mbytes. The file name and parameter name are not important.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.
domain	Name of the domain.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>
    </domain>
    <logo>
        <url></url>
    </logo>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
url*	The URL where the domain logo is located.

Example

The example below shows a cURL request that could be used, for example, in the BASH interpreter.

```
curl -F "file=@logo.jpg" -F "domain=example.com" -F "token=a1b2c3" 'https:///api/add_logo.xml'
```

This request adds a logo that is saved in the file logo.jpg for the domain example.com. The request also passes the access token a1b2c3 for this domain.

del_logo

This method is used for removing a domain logo.

Request syntax

`token`	The token that was obtained as the result of the `get_token` method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the `get_token` method.
`domain`	Name of the domain.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>
    </domain>
  </domains>
</action>
```

`success`	This element is present if the method was called successfully.
`error`	Error message. This element is included in the structure if errors occurred when the method was executed.
`name*`	Name of the domain.

get_domain_users

This method is used for getting a list of mailboxes.

The method returns a list of mailboxes in the domain that is associated with the token. The list is paginated, starting from the page with the specified number. Page numeration starts from 1.

Request syntax

`token`	Access token.

on_page	Number of mailbox records on a single page. The value of this parameter cannot be more than 100; by default, it is set to 100 records.
page	Page number to start the list of mailboxes. By default, the value is 1.

Response format

The method returns the following type of XML structure:

```
<page>
 <domains>
  <domain>
   <emails>
    <action-status/>
     <email><email-name>...</email-name></email>
     <email><email-name>...</email-name></email>
     ...
     <found>10</found>
     <total>100</total>
   </emails>
   <name>...</name>
   <status>added | mx-activate | domain-activate</status>
   <emails-max-count>100</emails-max-count>
  </domain>
 </domains>
</page>
```

action-status*	Error messages.
name*	Name of the domain.
email-name*	Name of the mailbox.
found*	Number of mailboxes to display on a single page.
status*	Domain status (added, mx-activate, domain-activate).
emails-max-count*	Maximum number of mailboxes allowed for this domain.

check_user

This method is used to verify the existence of a user.

Request syntax

token	Access token.
login	The user's login for accessing the mailbox.

Response format

The method returns an XML response:

- <result>exists</result> — User exists.
- <result>nouser</result> — User not found.

add_admin

This method is for adding an additional domain administrator.

The method adds an additional domain administrator. The additional administrator can get authenticated on the pdd.yandex.ru page, see all the domains added for him in his list of domains, and perform all operations with mailboxes on this domain.

Attention!
Be careful to type the correct login for the additional administrator, otherwise you may give some unknown person access to all your mailboxes. After calling the method, you must log out, then log in as the additional administrator and make sure that the domain appeared on his `http://pdd.yandex.ru` page.

Request syntax

`token`	The token that was obtained as the result of the `get_token` method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the `get_token` method.
`domain`	The name of the domain that the additional administrator is being added to.
`login`	The Yandex login name of the additional administrator.

Response format

The method returns the following type of XML structure:

```
<action>
  <domain>
   <status>
     <success/>
     <error>...</error>
   </status>
   <name>...</name>
   <new-admin> ... </new-admin>
  </domain>
</action>
```

`success`	This element is present if the method was called successfully.
`error`	Error message. This element is included in the structure if errors occurred when the method was executed.
`name*`	Name of the domain.
`new-admin*`	The name of the additional domain administrator.

del_admin

This method is for removing an additional domain administrator.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.
domain	The name of the domain that the additional administrator is being removed from.
login	The Yandex login name of the additional administrator.

Response format

The method returns the following type of XML structure:

```
<action>
  <domain>
   <status>
     <success/>
     <error>...</error>
   </status>
   <name>...</name>
   <new-admin> ... </new-admin>
  </domain>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
new-admin*	The name of the additional domain administrator.

get_admins

This method is for getting a list of additional domain administrators.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.
domain	The name of the domain to get a list of additional administrators for.

Response format

The method returns the following type of XML structure:

```
<action>
  <domain>
   <status>
     <success/>
     <error>...</error>
   </status>
   <name>...</name>
   <other-admins>
    <login>...</login>
   </other-admins>
  </domain>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
login*	The name of the additional domain administrator.

User accounts

Describes the Yandex.Mail for Domains API methods that are for performing operations with user accounts.

Method	Use
reg_user_token	User registration
reg_user_crypto	Creating a user with an encrypted password
reg_user	Creating a mailbox in a secondary domain
get_mail_info	Getting the number of unread messages
get_user_info	Getting user data
edit_user	Changing user data
set_forward	Setting up forwarding
get_forward_list	Getting a list of forwarding and filters
delete_forward	Removing forwarding and filters
delete_user	Removing a user
del_user	Removing a mailbox in a secondary domain

reg_user_token

This method is used for registering a user.

Request syntax

```
https://pddimp.yandex.ru/reg_user_token.xml ?
token=<access token>
 & u_login=<user login>
 & u_password=<user password>
```

`token` Access token.

`u_login` The user's login for accessing the mailbox.

`u_password` The user's password for accessing the mailbox.

Response format

If no errors occurred, the method returns `<ok uid="..."/>`; otherwise, it returns `<error reason='...'/>`.

reg_user_crypto

This method is for creating a user with an encrypted password.

Note:
Encryption does not affect how the password is entered to log in to the mailbox. The encrypted password is typed exactly as it was before it was encrypted.

Request syntax

token	Access token.
login	Name of the mailbox.
password	The encrypted version of the mailbox password, i.e. MD5-CRYPT hash of the password as a string with the following format: "1" + 8 symbols [a-zA-Z0-9./] + "$" + 22 symbols [a-zA-Z0-9./]

reg_user

This method is for creating a mailbox on a secondary domain.

The method works similarly to the reg_user_token method, except the user is not created on the current domain (meaning, not on the domain that is associated with the token), but on any domain that is connected to your account.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.
domain	Name of the domain.
login	Name of the mailbox.
passwd	User's password.
cryptopasswd	MD5-encrypted password for accessing the mailbox (similar to the reg_user_crypto method). If this parameter is set, the passwd parameter is ignored.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>

      <email>
        <name_email>...</name_email>
      </email>

    </domain>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.

name_email*	Mailbox name (login).

get_mail_info

This method is used for getting the number of unread messages.

Request syntax

token	Access token.
login	Name of the mailbox.

get_user_info

This method is used for getting user data.

Request syntax

token	Access token.
login	The user login that you are getting data for.

Response format

The method returns the following type of XML structure:

```
<page>
 <domain>
  <name></name>
  <user>
   <login>ivan</login>
   <birth_date>1900-01-01</birth_date>
   <fname>Ivanov</fname>
   <iname>Ivan</iname>
   <hinta></hinta>
   <hintq>Favorite food</hintq>
   <mail_format>...</mail_format>
   <charset></charset>
   <nickname>user</nickname>
   <sex>1</sex>
   <enabled>1</enabled>
   <signed_eula>1</signed_eula>
  </user>
 </domain>
</page>
```

login*	User's login name.
birth_date*	Date of birth in the format YYYY-MM-DD.
fname*	User's last name.
iname*	User's first name.
hinta*	Answer to the secret question.
hintq*	Secret question.

`mail_format*`	The preferred format for creating email messages.
`charset*`	Character encoding.
`nickname*`	The user's nickname.
`sex*`	User's gender: 0 – Not specified. 1 – Male. 2 – Female.
`enabled*`	Mailbox state: 1 – Enabled and receiving mail. 0 – Disabled and not accepting mail.
`signed_eula*`	Whether the user accepted the terms of use. 1 – Yes. 0 – No.

edit_user

This method is used for editing user data.

The method lets you change personal user data: first name, last name, gender, password, secret question, and the answer to the secret question.

Request syntax

`token`	Access token.
`login`	The user's login for accessing the mailbox.
`password`	User's password.
`domain_name`	User's mail domain name.
`iname`	User's first name.
`fname`	User's last name.
`sex`	User's gender: 0 – Not specified. 1 – Male. 2 – Female.
`hintq`	Secret question.
`hinta`	Answer to the secret question.

Response format

If no errors occurred, the method returns `<ok uid="..."/>`; otherwise, it returns `<error reason='...'/>`.

set_forward

This method is used for setting up forwarding for the user.

Request syntax

`token`	Access token.
`login`	The user's login for accessing the mailbox.
`address`	The email address that mail from the `login` mailbox will be forwarded to.
`copy`	This parameter determines whether to save copies of messages in the `login` mailbox after forwarding: “yes” — Save. “no” — Delete.

Response format

If no errors occurred, the method returns `<ok/>`; otherwise, it returns `<error reason='...'/>`.

get_forward_list

This method is used for getting a list of forwarding and filters.

Request syntax

`token`	Access token.
`login`	The user's login for accessing the mailbox.

Response format

If no errors occurred, the method returns `<ok uid="..."/>`«1»<ok><filters><filter>...</filter>...</filters></ok>«2»; otherwise, it returns `<error reason='...'/>`.

delete_forward

This method is used for removing forwarding or filters.

Request syntax

`token`	Access token.
`login`	The user's login for accessing the mailbox.
`filter_id`	ID of the filter to be deleted.

delete_user

This method removes a user.

Request syntax

token	Access token.
login	The user's login for accessing the mailbox.

del_user

This method is for deleting a mailbox in a secondary domain.

Request syntax

token	The token that was obtained as the result of the get_token method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the get_token method.
domain	Name of the domain.
login	Name of the mailbox.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>

      <email>
        <name_email>...</name_email>
      </email>

    </domain>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Domain name
name_email*	Mailbox name (login).

Distribution lists

Describes the Yandex.Mail for Domains API methods that are for creating and deleting general distribution lists for the domain.

Method	Use
create_general_maillist	Creating a general distribution list
delete_general_maillist	Removing a general distribution list

create_general_maillist

This method is for creating a general distribution list for the domain that includes all the users on the domain.

Request syntax

`token`	The token that was obtained as the result of the `get_token` method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the `get_token` method.
`domain`	Name of the domain.
`ml_name`	Name of a general distribution list.

Response format

The method returns the following type of XML structure:

```
<action>
  <domain>
   <status>
     <success/>
     <error>...</error>
   </status>
   <name>...</name>
   <users>
    <login>...</login>
   </users>
  </domain>
</action>
```

`success`	This element is present if the method was called successfully.
`error`	Error message. This element is included in the structure if errors occurred when the method was executed.
`name*`	Name of the domain.
`login*`	Name of the domain user.

delete_general_maillist

This method is for deleting the general distribution list for the domain that includes all the users on the domain.

Request syntax

`token`	The token that was obtained as the result of the `get_token` method. The method should only be called after connecting at least one domain "manually" and getting a token for it using the `get_token` method.
`domain`	Name of the domain.

Response format

The method returns the following type of XML structure:

```
<action>
  <domain>
   <status>
     <success/>
     <error>...</error>
   </status>
   <name>...</name>
   <users>
    <login>...</login>
   </users>
  </domain>
</action>
```

`success`	This element is present if the method was called successfully.
`error`	Error message. This element is included in the structure if errors occurred when the method was executed.
`name*`	Name of the domain.
`login*`	Name of the domain user.

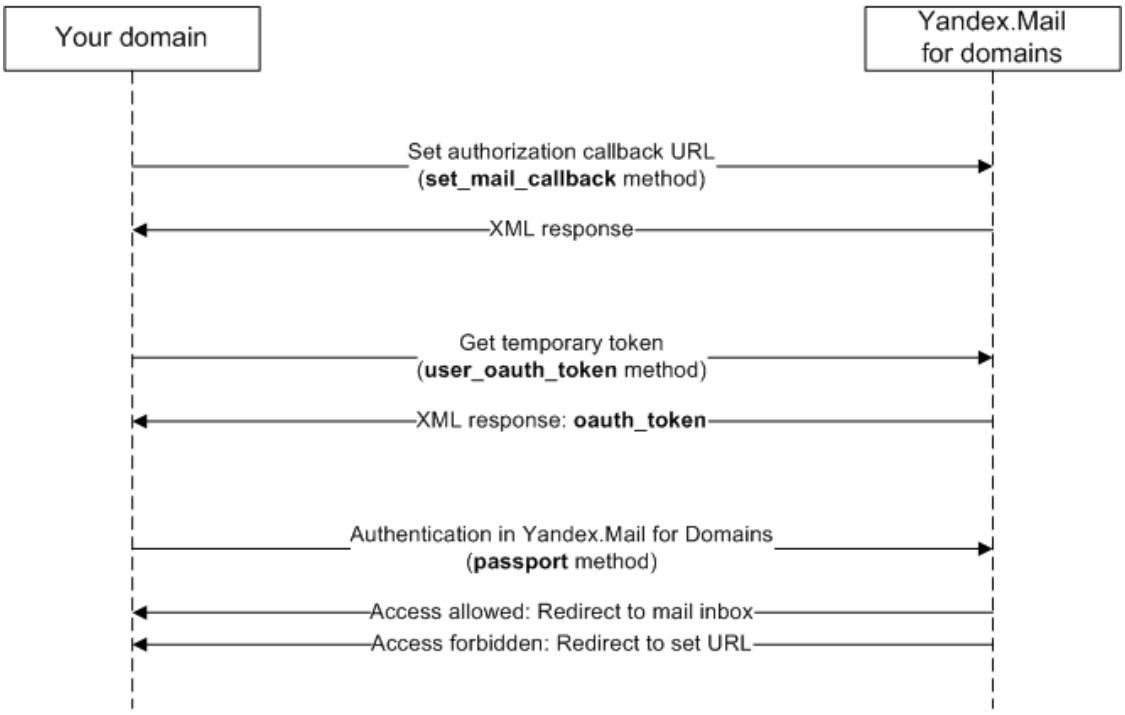
Authorizing users

Describes the Yandex.Mail for Domains API methods that are used for configuring authorization using a temporary OAuth token.

Method	Use
user_oauth_token	Getting a temporary access token
passport	Authorization using a temporary token
set_mail_callback	Setting an authorization callback URL

The methods implement the following authorization scenario in Yandex.Mail for Domains (see diagram):

1. Setting an authorization callback URL. Implemented using the set_mail_callback method. This method sets the authorization callback URL that the redirect goes to if an unauthorized user tries to access mail.
2. Getting a temporary access token. The authorization process itself uses a so-called temporary token, which is valid for 30 seconds. The temporary token is returned in the response to the user_oauth_token method.
3. Authorization. Implemented using the passport method. If access rights are verified, Yandex.Mail for Domains redirects the user's browser to the email inbox; otherwise, it redirects to the URL that is set in the parameters for this method.



The purpose and syntax of each of the methods is described in detail below, along with their usage.

user_oauth_token

This method is used for getting a temporary token for authorization.

The method returns a temporary token for authenticating in the Yandex.Mail web interface. The actual token authentication is performed by the passport method. The temporary token expires after 30 seconds.

Request syntax

token	The token that was obtained as the result of the get_token method.
domain	Name of the domain.
login	The user's login for accessing the mailbox.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>

      <email>
        <email_name>...</email_name>
        <oauth-token>...</oauth-token>
      </email>

    </domain>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
email_name*	Name of the mailbox (including the domain part) that the temporary token was created for.
oauth-token*	Temporary token.

passport

This method is for authenticating using a temporary token.

The method makes sure the temporary token is valid. If the token is valid, the method authenticates the user in the Yandex.Mail for Domains web interface and performs a redirect to the email inbox. If authentication is unsuccessful, a redirect is performed to the specified URL.

Request syntax

```
http://passport.yandex.ru/passport ? mode=oauth & type=trusted-pdd-partner
 & error_retpath=<redirect URL>
 & access_token=<access token>
http://passport.yandex.ru/passport ? mode=oauth & type=trusted-pdd-partner
 & error_retpath=<redirect URL>
 & access_token=<access token>
```

error_retpath

The URL to redirect to when authentication is unsuccessful. The URL should be correctly encoded using urlencode.

access_token	The token obtained as a result of the user_oauth_token method.

set_mail_callback

This method is used for setting an authorization callback URL.

Request syntax

token	The token that was obtained as the result of the get_token method.
domain	The domain of the user that the callback is being set for.
callback	The authorization callback URL. Must be correctly encoded using urlencode.

Response format

The method returns the following type of XML structure:

```
<action>
  <status>
    <success/>
    <error>...</error>
  </status>
  <domains>
    <domain>
      <name>...</name>

      <email>
        <email_name>...</email_name>
        <oauth-token>...</oauth-token>
      </email>

    </domain>
  </domains>
</action>
```

success	This element is present if the method was called successfully.
error	Error message. This element is included in the structure if errors occurred when the method was executed.
name*	Name of the domain.
email_name*	Name of the mailbox (including the domain part) that the temporary token was created for.
oauth-token*	Temporary token.

Using authorization methods

This section illustrates how to use authorization methods, using the example of instant email access directly from your web site.

You can put a link (such as “Mail”) on your web site so that users who are already authenticated on the site can follow the link directly to their mailboxes, without having to log in again. In this case, authentication on Yandex.Mail for Domains is performed automatically using the login and password that were entered when logging onto the site.

To provide instant email access through a link on your web site, you must:

1. Create mailboxes, if they have not already been created.
2. Get an access token.
3. Put a link for instant mailbox access on your web site.
4. Develop the authorization page.
5. Put the authorization page at the authorization URL.
6. Assign the authorization URL to your domain.
7. Develop the authorization module.
8. Install the authorization module on your web site.
9. Check that the link works using test accounts.

Recommendations for ensuring data security

You are completely responsible for the security of authorization on your web site. If trespassers are able to log in on your web site, they can also get access to a user's email using your link.

You are fully responsible for passing an authenticated user's login to Yandex.Mail for Domains. If you pass the login of a non-authenticated user without any additional authorization elements (for example, the password or the session ID in a cookie), it will be very easy for trespassers to get access to this user's email.

For protecting your users' personal data, we strongly recommend:

- Using safe authorization methods, in particular:
  - Send the login, password, and other authorization elements over HTTPS.
  - Generate authentication cookies that are long and difficult to guess.
  - Update the authentication cookie regularly (for example, once every few hours).
- Do not use so-called “convenient” authorization methods, which make it easier for trespassers to access email along with your users:
  - A “perpetual” cookie or “eternal” session. A user who did not log out of an authenticated session on your web site remains authenticated on the same computer forever, no matter how many times he goes to your web site.
  - Autologin links – links for authenticating in the email system automatically.

Long-lasting authenticated sessions and autologin links increase the risk of leaking personal data. So if your web site provides such “convenient” authorization methods, we do not recommend giving access to users who have a “perpetual” cookie or who used an autologin link. For example, if an autologin link is used, you can check the authentication cookie, and if there isn't one, you can ask the user to enter a password.

Please note that in the current implementation of Yandex.Mail for Domains, closing the authenticated session on the web site is not synchronized with closing the session in the mailbox, so the sessions end independently of each other. This means that when users log off your web site, they may still be logged in to their mailboxes, and vice versa.

Creating mailboxes

Mailboxes in Yandex.Mail for Domains are not created automatically when following the link that you put on your web site. So the first step is to check for the existence of mailboxes and create them as necessary for all the users on your web site. To register mailboxes, use the API methods described in the section User accounts:

- reg_user_token.
- reg_user_crypto.
- reg_user.

When registering mailboxes, we recommend providing real passwords, since users can get to their email not only from the link on your web site, but also using mail clients (including mobile clients). We recommend synchronizing passwords for existing mailboxes with the current passwords of your site users, via the API.

In the future, you should keep track of new users who register on your web site and create mailboxes for them on Yandex.Mail for Domains. You will also need to delete the mailboxes of any users who delete their accounts on your web site (the delete_user and del_user methods).

Link for instant mailbox access

You should place a link on your web site in the format

```
<a href="http://mail.yandex.ru/for/domain.ru?
loginfrompartner=user@domain.ru">Mail</a>
```

where user@domain.ru must be replaced with the email address of a user who is authenticated on your web site.

Clicking this link leads to verification of the user's authorization, which is performed automatically in Yandex.Mail for Domains. If access rights are verified, the user is redirected to the mailbox.

Authorization page

You must develop a page for authorizing users in Yandex.Mail for Domains. This page must have fields for entering the login name and password for accessing a mailbox. Additionally, you must bind the authorization module to this page for authenticating users in Yandex.Mail for Domains.

The authorization page resides on your web site and we will refer to its URL as the authorization URL for your users to access mailboxes.

Assigning the authorization URL to the domain

Both authorized and unauthorized users may try to get access to email. If a user is not authenticated on your web site and tries to enter Yandex.Mail for Domains using the “Mail” link, this user will be redirected to the login page for Yandex.Mail for Domains. To avoid this, you should assign the authorization URL to the domain using the set_mail_callback method. In this case, if a user who is not authenticated on your web site tries to access email, this user will be redirected to the authorization page that you specify. The URL of this page is passed in the callback parameter of the set_mail_callback method.

Authorization module

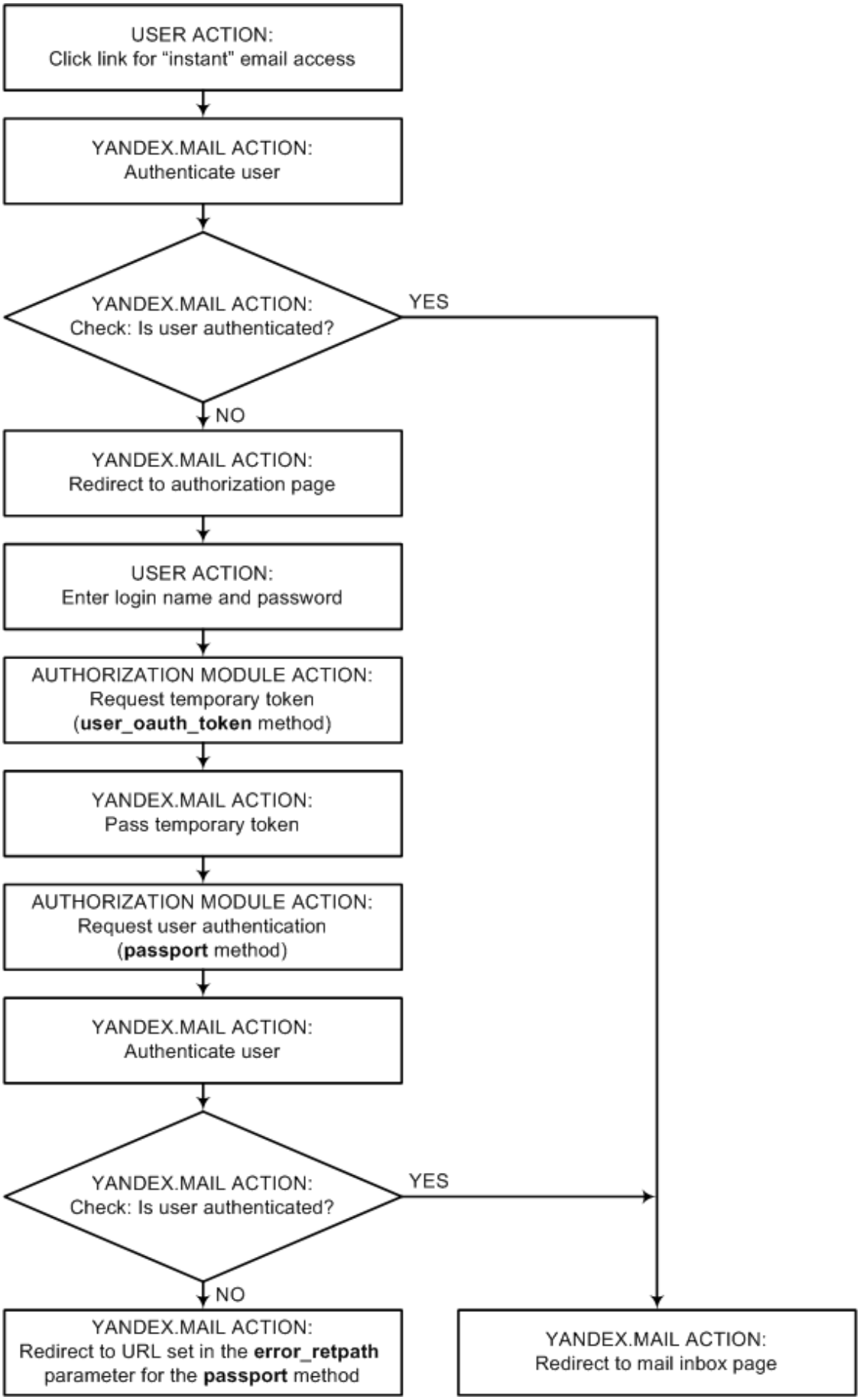
Automatic authorization in Yandex.Mail for Domains API is implemented by a special module. This module resides on your resource and performs all operations necessary for verifying email access rights.

The authorization module is developed individually for each web site, and can be implemented in various ways (for example, as a script). To develop it, you will need programming skills in the language that your web site is written in (for example, PHP or Perl). If you do not have these skills, we recommend contacting a professional with the appropriate experience (for example, your web site developer).

The authorization module must perform the following actions:

- Check for the user's authentication on the web site (for example, using a cookie).
- Identify the login name of the authenticated user.
- Get a temporary token using the user_oauth_token method.
- Authenticate the user in Yandex.Mail for Domains using the passport method.

This module implements the following authorization scenario in Yandex.Mail for Domains (see diagram):



When a user clicks the link for instant email access, an attempt is made to access the mailbox. The login and domain name are passed from your web site to Yandex.Mail for Domains, which attempts to authenticate the user with this login. If access rights are verified, the user's browser is redirected to the inbox page.

If the user could not be authenticated, Yandex.Mail for Domains redirects the browser to the authorization page. On this page, the user enters a login name and password, and then your resource executes the authorization module.

The authorization module calls the user_oauth_token method, passing it a parameter that contains the login (which is the mailbox name), along with the domain name. In response to the user_oauth_token method, a temporary token is returned, which will be used for authorization. The temporary token expires after 30 seconds, so all further operations must be completed within this time.

Note:
Don't confuse the temporary token with the access token. The access token does not expire, and it is used for accessing all of the API methods. The temporary token is only valid for 30 seconds, and it is needed only for passing the fact of a particular user's authentication on your web site to Yandex.Mail for Domains.

The authorization module calls the passport method, passing the temporary token to it as a parameter. If the token is valid, Yandex.Mail for Domains performs a browser redirect to the inbox page. Any other situation indicates that an error occurred, and the user cannot be authenticated. In this case, Yandex.Mail for Domains performs a browser redirect to the URL that is specified in the error_retpath parameter of the passport method.

Putting the authorization module on your web site

The authorization module you have developed must be installed on your web site and bound to the authorization URL. Binding it must ensure that the module will be executed after the browser redirect to the authorization page so the user can enter the login and password.

Checking how automatic authorization works

We strongly recommend checking how the instant email access link works on test accounts, to make sure that any errors in the authorization module do not interfere with normal use of your web site.

If you are sure that you have done everything correctly, but authentication is still not occurring or our methods are returning errors, contact Support using the feedback form. For efficient problem solving, we recommend providing the following information in your message:

- Your domain and the mailbox name that authentication is not working on.
- The URL of the web site where you have added a link to instant email access.
- The API methods that you called or performed a redirect on.
- The responses that were returned by the methods you used.
- The URL or, if possible, a screenshot of the page that you or your users saw after clicking the link for instant email access.
- The source code of your authorization module, if possible.

Importing email

Describes the Yandex.Mail for Domains API methods that are used for managing mail import.

Method	Use
set_domain	Saving import settings for a domain
start_import	Starting mailbox import
check_import	Checking the status of the mail import process
stop_import	Stopping mail import
reg_and_imp	Registering a user and starting mail import
import_imap	Starting folder import over IMAP

set_domain

This method is for saving import settings for the domain.

Before starting the import process, save the import settings for your domain: the POP3 server address and port (if a non-standard port is used).

Request syntax

`token` Access token.

`method` Protocol: POP3 or IMAP.

`ext_serv` Domain name of the POP3 or IMAP source server.

`ext_port` This parameter is specified if the port number is not the standard one used for the specified protocol.

Standard ports:

110 – POP3 without SSL

995 – POP3 with SSL

`isssl` This parameter defines whether the server supports SSL. If the connection is made over SSL, the parameter is omitted. Other, the value “no” needs to be specified.

`callback` If the parameter is not empty, when the user mailbox import finishes, an HTTP request is made to this address with the parameter `login="imported user's login"`.

If the import finishes without errors, the request returns the XML structure: `<page><status>moved</status></page>`.

Response format

If no errors occurred, the method returns `<ok/>`; otherwise, it returns `<error reason='...'/>`.

start_import

This method is for starting mailbox import.

Request syntax

`token`	Access token.
`login`	Name of the mailbox.
`ext_login`	The user's login name on the source server. If the login on the source server matches the login for the mailbox being created, the parameter can be omitted.
`password`	The user's password on the source server.

check_import

This method is used for checking the mail import status.

Request syntax

`token`	Access token.
`login`	Login name of a new Yandex user; specify only the part of the address before the “@” symbol.

stop_import

This method is for stopping mailbox import.

Request syntax

`token`	Access token.
`login`	Name of the mailbox.

reg_and_imp

This method is used for registering a user and importing the user's mail.

Request syntax

`token`	Access token.
`login`	The user login for accessing the mailbox that is being created.
`ext_login`	The user's login name on the source server. If the login on the source server matches the login for the mailbox being created, the parameter can be omitted.

`inn_password`	The user's Yandex password.
`ext_password`	The user's password on the source server.
`fwd_email`	If this parameter is set, the mailbox will have messages forwarded to the email address specified in this parameter, for example, fwdaddress@somedomain.name.
`fwd_copy`	This parameter makes sense only if `fwd_email` is set. If `fwd_copy` is set to 0, copies of messages are not saved in this mailbox after forwarding. Otherwise, copies of messages are saved in the current mailbox.

Response format

If no errors occurred, the method returns `<ok/>`; otherwise, it returns `<error reason='...'/>`.

import_imap

This method is used for importing a single folder over IMAP.

The method starts import of one of the user's IMAP folders. If the user does not exist, he is created, and a password is set for him. Old messages are not removed from the folder, so repeating the import will copy all the messages to the folder again.

Request syntax

`token`	Access token.
`login`	Name of the mailbox.
`ext_login`	The user's login name on the source server. If the login on the source server matches the login for the mailbox being created, the parameter can be omitted.
`ext_password`	The user's password on the source server.
`int_password`	The user's password for Yandex.Mail for Domains. This parameter is set only if a new mailbox is being created during import. If the mailbox already exists, the `int_password` parameter is ignored.
`copy_one_folder`	The name of the folder that needs to be imported over IMAP. If the folder name contains Cyrillic characters, it should be passed in UTF-8 (and URL encoded).

Managing DKIM

Describes the Yandex.Mail for Domains API methods that are used for managing DKIM use.

DKIM (DomainKeys Identified Mail) is a way to sign email messages so that the sender's domain can be identified. This type of signature also reduces the chances of the message being flagged as spam.

You can use the Yandex.Mail for Domains API to enable and disable the signature on messages being sent from your domain.

Method	Use
enable	Enabling DKIM.
status	Getting DKIM status.
disable	Disabling DKIM.

enable

Use this method to enable DKIM usage for the domain.

Request syntax

`token`	Access token.
`domain`	The name of the domain to enable DKIM for.

Response format

This XML document contains all the elements that can possibly be found in the response, including those that are mutually exclusive. Elements are shown with sample values.

```
<domains>
  <action-status>
    <success/>
    <error>ERROR 403: Forbidden.</error>
  </action-status>
  <domain>
    <name>example.com</name>
    <dkim>
      <enabled>yes|no</enabled>
      <txtrecord>mail._domainkey IN TXT "v=DKIM1; k=rsa; t=s; p=MIGfMA0GCS//
EBtaCOteH4EBqJlKperJ
+5BPEGS7N3fFkdeKllShrM73nm4xPdZmt2jNnmgWMeQySGYW5VUJ8PCePanwIXcW8YnqS7zw+grL/
PHhUtf3ofSLmtVM3rSWmJ9qHFhxWmPFplPe5OsvpO+fphiMOrTnzzV/004S/jQIDAQAB" ;
DKIM key mail for example.com</txtrecord>
    </dkim>
  </domain>
</domains>
```

`action-status`

Status of request processing. The nested element depends on the processing source:

- `<success/>` — Operation completed successfully.
- `<error>` — Operation completed with an error. The error description is in the element's value. Possible errors:

Error description	Comments
ERROR 400: Bad Request.	A mandatory parameter was omitted, or a non-existing domain was specified.
ERROR 403: Forbidden.	An invalid access token was passed.

`domain` The domain to perform the operation on.

`name` Name of the domain.

`enabled` Status of DKIM options. Acceptable values:

- yes — Email signature feature enabled.
- no — Feature disabled.

`txtrecord` A TXT record with an open DKIM key for independently making settings. The record consists of three parts:

- The record name: `mail_domainkey`
- DKIM parameters:

```
v=DKIM1;
k=rsa;
t=s;
p=MIGfMA0GCS//EBtaCOteH4EBqJlKperJ
+5BPEGS7N3fFkdeKllShrM73nm4xPdZmt2jNnmgWMeQySGYW5VUJ8PCePanwIXcW8YnqS7
zw+grL/PHhUtf3ofSLmtVM3rSWmJ9qHFhxWmPFplPe5OsvpO+fphiMOrTnzzV/004S/
jQIDAQAB
```

  The `p` parameter must contain the DKIM open key.

- Reference to the Yandex.Mail for Domains domain: `DKIM key mail for example.com`

status

Use this method for checking the DKIM status of the domain.

Request syntax

`token` Access token.

`domain` Name of the mailbox.

Response format

This XML document contains all the elements that can possibly be found in the response, including those that are mutually exclusive. Elements are shown with sample values.

```
<domains>
  <action-status>
    <success/>
    <error>ERROR 400: Bad Request.</error>
  </action-status>
  <domain>
    <name>example.com</name>
    <dkim>
      <enabled>yes</enabled>
      <txtrecord>mail._domainkey IN TXT "v=DKIM1; k=rsa; t=s; p=MIGfMA0GCS//
EBtaCOteH4EBqJlKperJ
+5BPEGS7N3fFkdeKllShrM73nm4xPdZmt2jNnmgWMeQySGYW5VUJ8PCePanwIXcW8YnqS7zw+grL/
PHhUtf3ofSLmtVM3rSWmJ9qHFhxWmPFplPe5OsvpO+fphiMOrTnzzV/004S/jQIDAQAB" ; DKIM key
mail for example.com</txtrecord>
      <nsready>yes</nsready>
      <mailready>yes</mailready>
    </dkim>
  </domain>
</domains>
```

`action-status`

Status of request processing. The nested element depends on the processing source:

- `<success/>` — Operation completed successfully.
- `<error>` — Operation completed with an error. The error description is in the element's value. Possible errors:

Error description	Comments
ERROR 400: Bad Request.	A mandatory parameter was omitted, or a non-existing domain was specified.
ERROR 403: Forbidden.	An invalid access token was passed.

`domain` The domain to perform the operation on.

`name` Name of the domain.

`enabled` Status of DKIM options. Acceptable values:

- yes — Email signature feature enabled.
- no — Feature disabled.

`txtrecord` A TXT record with an open DKIM key for independently making settings. The record consists of three parts:

- The record name: `mail_domainkey`
- DKIM parameters:

  ```
  v=DKIM1;
  k=rsa;
  t=s;
  p=MIGfMA0GCS//EBtaCOteH4EBqJlKperJ
  +5BPEGS7N3fFkdeKllShrM73nm4xPdZmt2jNnmgWMeQySGYW5VUJ8PCePanwIXcW8YnqS7
  zw+grL/PHhUtf3ofSLmtVM3rSWmJ9qHFhxWmPFplPe5OsvpO+fphiMOrTnzzV/004S/
  jQIDAQAB
  ```

  The `p` parameter must contain the DKIM open key.
- Reference to the Yandex.Mail for Domains domain: `DKIM key mail for example.com`

`nsready` Flag that a TXT record with a DKIM open key was added for the domain. Acceptable values:

- yes — TXT record is installed.
- no — TXT record is missing.

For delegated domains, the necessary DNS records are added automatically.

`mailready` Flag for whether Yandex.Mail for Domains is ready to sign email using DKIM. Acceptable values:

- yes — Yandex.Mail for Domains adds signatures to email being sent from this domain.
- no — Yandex.Mail for Domains is not ready to sign emails for this domain.

  Yandex.Mail for Domains updates DKIM data with some delay. If the "ready" flag is still negative, but all the conditions for enabling DKIM have been met, you just need to wait.

disable

Use this method to disable DKIM usage for the domain.

Request syntax

`token` Access token.

`domain` Login name of a new Yandex user; specify only the part of the address before the “@” symbol.

Response format

This XML document contains all the elements that can possibly be found in the response, including those that are mutually exclusive. Elements are shown with sample values.

```
<domains>
  <action-status>
    <success/>
    <error>ERROR 400: Bad Request.</error>
  </action-status>
  <domain>
    <name>example.com</name>
  </domain>
</domains>
```

`action-status`

Status of request processing. The nested element depends on the processing source:

- `<success/>` — Operation completed successfully.
- `<error>` — Operation completed with an error. The error description is in the element's value. Possible errors:

Error description	Comments
ERROR 400: Bad Request.	A mandatory parameter was omitted, or a non-existing domain was specified.
ERROR 403: Forbidden.	An invalid access token was passed.

`domain` The domain to perform the operation on.

`name` Name of the domain.

DNS API

Describes the methods of the Yandex.Mail DNS API.

The Yandex.Mail for Domains DNS API makes it possible to administer the DNS for your domain if you delegate your domain to Yandex.

The API is an HTTP(S) interface. Parameters are passed in the request string or request body. All parameters should be passed using urlencode encoding. The result is output in the response body.

The table below lists the methods and their use.

Method	Use
get_token	Getting access
add_a_record	Creating A records
add_aaaa_record	Creating AAAA records
add_cname_record	Creating CNAME records
add_mx_record	Creating MX records
add_ns_record	Creating NS records
add_srv_record	Creating SRV records
add_txt_record	Creating TXT records
get_domain_records	Reading records for the domain's zone
edit_a_record	Modifying A records
edit_aaaa_record	Modifying AAAA records
edit_cname_record	Modifying CNAME records
edit_mx_record	Modifying MX records
edit_ns_record	Modifying NS records
edit_srv_record	Modifying SRV records
edit_soa_record	Modifying SOA records
edit_txt_record	Modifying TXT records
delete_record	Deleting records

get_token

This method is used for getting an access token.

The access token is used for authentication in the DNS API, which is required for accessing the other API methods. You only need to get the token once.

To get a token, you should have a domain that is already signed up, authenticate as the domain administrator, and go to the address shown below.

Request syntax

`domain_name` The name of the domain that the access token is being requested for.

Example

Getting a token for the domain `example.com`:

Response format

If there are no errors, the method returns the following XML response:

```
<page>
  <ok token="d604468e74ffce4cb31ceef915a8739a3a5cb9dec"/>
</page>
```

XML response with errors:

```
<page>
  <error reason="no token found"/>
</page>
```

add_a_record

This method is used for creating an A record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
`content`	IP address for the record. Set in IPv4 standard format (for example, 194.84.46.241).

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains "ok"; otherwise, it contains an error message.

add_aaaa_record

This method is used for creating an AAAA record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.

subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
content	IP address for the record. Set in IPv6 standard format (for example, 2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d).

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains "ok"; otherwise, it contains an error message.

add_cname_record

This method is used for creating a CNAME record.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being added to or modified for.
subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
content	Full domain name, such as example.com.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains "ok"; otherwise, it contains an error message.

add_mx_record

This method is used for creating an MX record.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being added to or modified for.
subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
content	Full domain name, such as example.com.
priority	Priority of the record. The lower the value, the higher the priority. By default, the parameter value is set to 10.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains "ok"; otherwise, it contains an error message.

add_ns_record

This method is used for creating an NS record.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being added to or modified for.
subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
content	Full domain name, such as example.com.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

add_srv_record

This method is used for creating an SRV record.

Request syntax

The SRV record is formed from the values of the passed parameters. For example, the record "5 0 5269 xmpp.yandex.ru" consists of the following parts:

- 5 — priority of the record (`priority` parameter).
- 0 — weight of the record (`weight` parameter).
- 5269 — port the service is on (`port` parameter).
- xmpp.yandex.ru. — name of the domain the service is on (`target` parameter).

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
`priority`	Priority of the record. The lower the value, the higher the priority. By default, the parameter value is set to 10.
`weight`	The weight of the SRV record in relation to other SRV records for the same domain and with the same priority.
`port`	The TCP or UDP port of the host where the service resides.
`target`	The canonical name of the host providing the service.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

add_txt_record

This method is used for creating a TXT record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
`content`	Text for the TXT record. For example, "v=spf1 redirect=_spf.yandex.ru".

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

get_domain_records

This method is used for reading the domain's zone records.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that records are being read for.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
      <response>
        <record domain="yourdomain.ru" priority="" ttl="21600" subdomain="www"
type="A" id="342432432">127.0.0.1</record>
      </response>
      <nsdelegated/>
    </domain>
    <error>ok</error>
</domains></page>
```

`name`	Name of the domain that records were requested for.
`record`	The address specified in the DNS record. The record's characteristics are specified in the element's attributes.
`priority`	Priority of the DNS record.
`ttl`	Lifetime of the record, in seconds.
`subdomain`	Subdomain name.
`type`	Type of DNS record.
`id`	ID of the DNS record.
`nsdelegated`	Flag for a domain delegated to Yandex. If the domain is not delegated, this element is not included in the response.
`error`	If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

edit_a_record

This method is used for modifying an A record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`record_id`	ID of the record. The ID is contained in the value of the `id` attribute for the `record` element, which is returned by the `get_domain_records` method.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
`content`	IP address for the record. Set in IPv4 standard format (for example, 194.84.46.241).

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

edit_aaaa_record

This method is used for modifying an AAAA record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`record_id`	ID of the record. The ID is contained in the value of the `id` attribute for the `record` element, which is returned by the `get_domain_records` method.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
`content`	IP address for the record. Set in IPv6 standard format (for example, 2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d).

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

edit_cname_record

This method is used for modifying a CNAME record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`record_id`	ID of the record. The ID is contained in the value of the `id` attribute for the `record` element, which is returned by the `get_domain_records` method.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).

content	Full domain name, such as example.com.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains “ok”; otherwise, it contains an error message.

edit_mx_record

This method is used for modifying an MX record.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being added to or modified for.
subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
record_id	ID of the record. The ID is contained in the value of the id attribute for the record element, which is returned by the get_domain_records method.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
content	The IP address that should be specified in the DNS record.
priority	Priority of the record. The lower the value, the higher the priority. By default, the parameter value is set to 10.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains “ok”; otherwise, it contains an error message.

edit_ns_record

This method is used for modifying an NS record.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being added to or modified for.
subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
record_id	ID of the record. The ID is contained in the value of the id attribute for the record element, which is returned by the get_domain_records method.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
content	The IP address that should be specified in the DNS record.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains "ok"; otherwise, it contains an error message.

edit_srv_record

This method is used for modifying an SRV record.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being added to or modified for.
subdomain	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as example.com). For example, if subdomain=www, the record for http://www.example.com is created or edited. When subdomain=mail , it is for mail.example.com.
record_id	ID of the record. The ID is contained in the value of the id attribute for the record element, which is returned by the get_domain_records method.
ttl	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
priority	Priority of the record. The lower the value, the higher the priority. By default, the parameter value is set to 10.

`weight`	The weight of the SRV record in relation to other SRV records for the same domain and with the same priority.
`port`	The TCP or UDP port of the host where the service resides.
`target`	The canonical name of the host providing the service.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

edit_soa_record

This method is used for modifying an SOA record.

Request syntax

The new SOA record value is formed from the values of the passed parameters. For example, the record "ns1.yandex.ru. sysadmin.yourdomain.ru. 2012122006 600 300 2592000 900" consists of the following parts:

- ns1.yandex.ru. — The DNS server (the Yandex.Mail for Domains API automatically detects the DNS server).
- sysadmin.yourdomain.ru. — The email address of the domain administrator (the `admin_mail` parameter).
- 2012122006 — The version of the SOA record (the Yandex.Mail for Domains API automatically determines the version).
- 600 — Time interval for updating secondary DNS servers (`refresh` parameter).
- 300 — Time interval for sending repeated attempts to the primary server (`retry` parameter).
- 2592000 — Life of the zone records (`expire` parameter).
- 900 — How long to cache a negative response from the service (`neg_cache` parameter).

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
`admin_mail`	The email address of the domain administrator. Displayed in the SOA record.
`refresh`	Time between zone record updates on the slave DNS servers.
`retry`	Time between repeat attempts of the slave DNS servers to get the zone records (if the master server did not return anything).
`expire`	Timeout, after which the slave DNS server will consider the zone records non-existent (if the master server continues to not respond).
`neg_cache`	How long to cache a negative response from the DNS server (meaning the response `ERROR = NXDOMAIN` — no such record).

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

edit_txt_record

This method is used for modifying a TXT record.

Request syntax

`token`	The API user's access token.
`domain`	The name of the domain that the DNS record is being added to or modified for.
`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`record_id`	ID of the record. The ID is contained in the value of the `id` attribute for the `record` element, which is returned by the `get_domain_records` method.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — “21600”).
`content`	Text for the TXT record. For example, "v=spf1 redirect=_spf.yandex.ru".

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains “ok”; otherwise, it contains an error message.

delete_record

This method is used for deleting records.

Request syntax

token	The API user's access token.
domain	The name of the domain that the DNS record is being deleted for.
record_id	ID of the record. The ID is contained in the value of the id attribute for the record element, which is returned by the get_domain_records method.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the error element contains “ok”; otherwise, it contains an error message.

Request specification

This section describes the parameters for API method calls and the response format.

Method parameters

content*	The IP address that should be specified in the DNS record.
admin_mail*	The email address of the domain administrator. Displayed in the SOA record.
domain*	The name of the domain that the DNS record is being added to or modified for.
domain_name*	The name of the domain that the access token is being requested for.
expire*	Timeout, after which the slave DNS server will consider the zone records non-existent (if the master server continues to not respond).
neg_cache*	How long to cache a negative response from the DNS server (meaning the response ERROR = NXDOMAIN — no such record).
port*	The TCP or UDP port of the host where the service resides.
priority	Priority of the record. The lower the value, the higher the priority. By default, the parameter value is set to 10.
record_id	ID of the record. The ID is contained in the value of the id attribute for the record element, which is returned by the get_domain_records method.
refresh	Time between zone record updates on the slave DNS servers.
retry	Time between repeat attempts of the slave DNS servers to get the zone records (if the master server did not return anything).

`subdomain`	Subdomain name. This parameter is necessary if a record needs to be created or edited for a subdomain, but not for the domain itself (such as `example.com`). For example, if `subdomain=www`, the record for `http://www.example.com` is created or edited. When `subdomain=mail`, it is for `mail.example.com`.
`target`*	The canonical name of the host providing the service.
`token`*	The API user's access token.
`ttl`	Length of the record's life in seconds (if omitted, the default value will be used — "21600").
`weight`*	The weight of the SRV record in relation to other SRV records for the same domain and with the same priority.

Response format

The method returns an XML response in this format:

```
<page>
  <domains>
    <domain>
      <name>example.com</name>
    </domain>
    <error>[processing status]</error>
  </domains>
</page>
```

If no errors occurred, the `error` element contains "ok"; otherwise, it contains an error message.

API for hosters, registrars and domain parkers

The interface for interacting with the Yandex.Mail for Domains service documented here is designed for hosters, registrars and domain parkers.

The API allows hosting providers, registrars and domain parkers (further referred to as "registrars") to automatically sign up their users' domains for the Yandex.Mail for Domains service (with a single click) and import their email to Yandex.

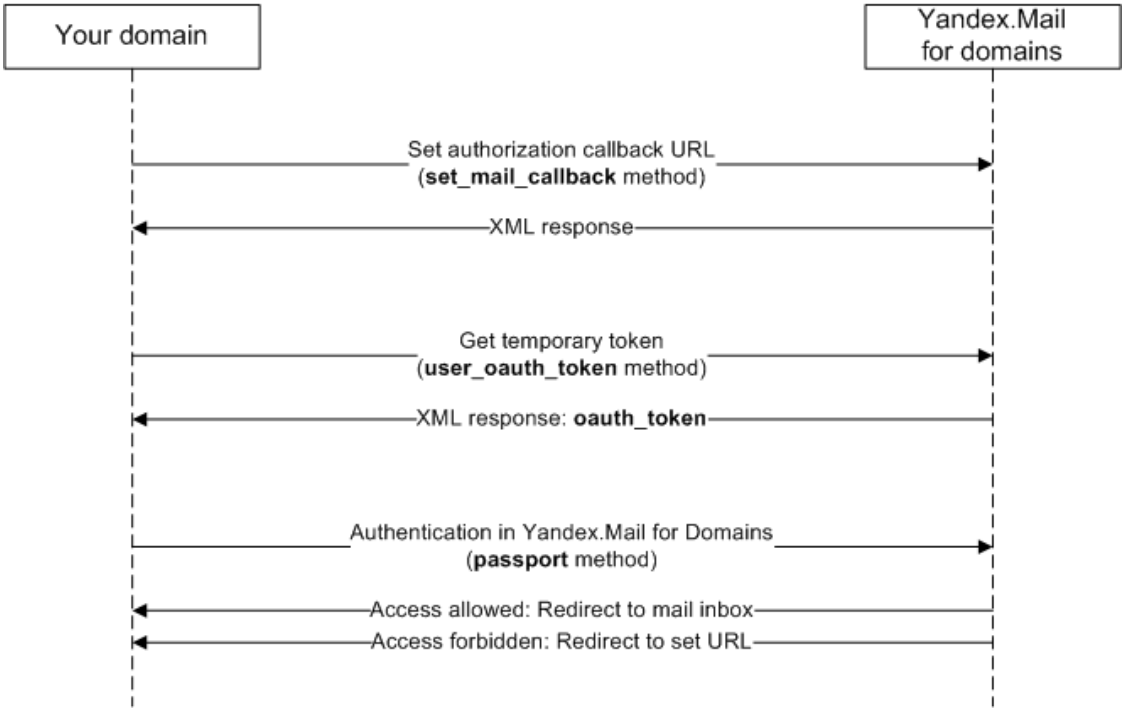
To do this, you need to:

1. Get a registrar ID.
2. Set callback URLs.
3. Make a request to sign up for Mail for Domains.
4. Authenticate.
5. Confirm domain ownership.
6. Get confirmation of domain sign-up.
7. Import user email.

In addition, the methods in this API let you get the following information:

- Current state of mail import.
- Current state of the domain.
- Notification of domain disconnection.

The illustration below shows the interaction between a registrar and Mail for Domains.



The registrar and Mail for Domains use HTTP methods to interact with each other.

The table lists the methods and their use.

Method	Use
Methods called on the registrar side	

Method	Use
create_registrar	Getting a registrar ID
save_registrar	Setting callback URLs
registraradd/do	Request to sign up for Mail for Domains
Importing email	Importing email
registrar_check_import	Checking the state of mail import
registrar_check_domain	Getting information about the current domain state
Methods called on the Mail for Domains side	
check_payed_method	Domain authentication
added_init_method	Confirmation of domain ownership
added_method	Confirmation of domain sign-up
transfer_succeed	Confirmation of email import
delete_method, something_delete	Notification of domain disconnection

Getting a registrar ID

The `create_registrar` method is used for getting the registrar ID.

The method is called on the registrar's side. To get an ID, you need to authenticate as a Yandex user and call the following URL from the same browser window:

`name`	Name of the registrar.
`password`	The password for accessing the API. The same password is used for calling other API methods.

Response format

In response to the method call, the following type of XML structure is returned:

```
<registrar>
  <error>ok|...</error>
  <id>...</id>
  <name>...</name>
</registrar>
```

`error`*	Request results. If the method was called successfully, the element contains “ok”; otherwise, it contains an error message.
`id`*	Unique registrar's ID.
`name`*	The registrar's name.

Setting callback URLs

The `save_registrar` method is used for setting callback URLs.

The Yandex server invokes callback URLs for a number of different events:

- Getting a request to sign up for Mail for Domains.
- Getting a response to the authentication request.
- Confirmation of domain ownership.
- Domain deletion.
- Completion of email import.

The URLs are set like this:

`registrar_id`	The registrar ID that was obtained using the `create_registrar` method.
`password`	The registrar's password. The password should be at least 8 symbols long.
`payed_url`	A URL for checking that the user really initiated signing up his domain for Yandex.Mail via your web interface.
`added_init`	The URL that is called on the Mail for Domains side if the domain status has successfully changed to “waiting for confirmation” (see below).
`added`	The URL that is called on the Mail for Domains side if the domain has been successfully added, confirmed and activated. When this URL is called, the user can already create or migrate mailboxes on this domain and use the full features of Yandex.Mail on this domain.
`delete_url`	The URL that is called on the Mail for Domains side if your user deletes the domain from Mail for Domains.
`transfer_succeed`	The URL that is called on the Mail for Domains side if the user's mailboxes were successfully migrated.

Response format

In response to the method call, the following type of XML structure is returned:

```
<registrar>
  <error>ok|...</error>
</registrar>
```

`error*`	Method call result. If executed successfully, contains “ok”; otherwise, contains an error message.

Request to sign up for Mail for Domains

The `registraradd/do` method is used for making a request to sign up a domain for Mail for Domains.

The request is executed when the "Sign up a domain" button is clicked:

```
https://pdd.yandex.ru/domains/domain/registraradd/do ?
domain=<domain name>
 & registrar_id=<registrar's ID>
 & service_id=<service ID>
 & payed_url=<authentication URL>
https://pdd.yandex.ru/domains/domain/registraradd/do ?
domain=<domain name>
 & registrar_id=<registrar's ID>
 & service_id=<service ID>
 & payed_url=<authentication URL>
```

domain	Name of the domain.
registrar_id	The registrar ID that was obtained using the create_registrar method.
service_id	Unique number generated by the registrar. Each time the "Sign up a domain" button is clicked, a new service_id is generated. For the service_id, use a number from 0 to 4294967295. Do not use numbers that are easy to guess for the service_id, such as a series of numbers in order. This service_id is used to authenticate the domain.
payed_url	A URL for checking that the user really initiated signing up his domain for Mail for Domains via your web interface.

On the Yandex page, the user enters his Mail for Domains login and password; after this, the check_payed_method method is called on the Mail for Domains side.

Domain authentication

The check_payed_method method is for making an authentication request to the domain.

The authentication request is called on the Mail for Domains side, making it possible to check whether the domain is paid for and whether the user wants to sign up.

This verification is made in the following way: When the registraradd/do method is called, the registrar generates a service_id and saves the service_id + domain pair (see the section Request to sign up for Mail for Domains). When executing the authentication request, the Mail for Domains service uses the check_payed_method method to pass the service_id and domain name. The registrar accepts the domain name and service_id, compares them to the saved data, and if it matches, sends confirmation of successful authentication to the Mail for Domains service.

If the check_payed_method request returned an error response, returned nothing, or timed out, the domain will not be signed up.

To call the method, the following URL must be specified in the payed_url parameter of the save_registrar method: https://registrar.domain.name/check_payed_method.

Attention!
If authentication is successful, the domain is not connected immediately, but is just added in the "waiting for confirmation" state. Fully-functional use of Mail on this domain will be available sometime later, when confirmation of domain ownership and MX configuration is received from the registrar (basically, the API is used for doing all the work that the user would have done manually if he had connected the domain himself, instead of using the button on your site). For more details, see the section Confirmation of domain ownership.

The check_payed_method method can be called over HTTP or HTTPS.

```
http(s)://registrar.domain.name/check_payed_method ?
domain=<domain name>
 & service_id=<service ID>
 & (password=<password> | sign=<signature>)
http(s)://registrar.domain.name/check_payed_method ?
domain=<domain name>
 & service_id=<service ID>
 & (password=<password> | sign=<signature>)
```

`domain`	Name of the domain.
`service_id`	A unique number that is passed when calling `registraradd/do`.
`password`	The password for accessing the API.
`sign`	`sign = sha1 (service_id + domain + password)`. Only passed in the HTTP version of the method.

Attention!
The `check_payed_method` method has a 10-second timeout, since it is called in the context of generating a user page. Users usually start to worry after a 3-second delay. For this reason, we don't recommend performing lengthy blocking operations when executing `payed_url` (if registering MX records requires such operations). It's better to perform these operations after the response to the `payed_url` HTTP request.

Response format

In response to the method call, the following type of XML structure is returned:

```
<is_payed>
 <value>True|False</value>
 <error>...</error>
</is_payed>
```

`value`*	Authentication results: "True" – The domain can be connected to Mail for Domains. In this case the registrar should configure an MX record for the domain that points to mx.yandex.ru with priority 10, and delete all existing MX records. "False" — Negative. If an error occurred, the `value` element does not contain any data.
`error`*	Request results. If the method was called successfully, the element contains "ok"; otherwise, it contains an error message.

Confirmation of domain ownership

Confirmation of domain ownership is made using the `added_init_method` method.

Confirmation of domain ownership is necessary in order to avoid any errors on your side when transferring domains. Theoretically, Yandex could get a request for signing up a random domain. If such a domain were signed up automatically, the consequences would be that the real domain owner would not be able to sign it up.

The `added_init_method` method is called on the Yandex side if a positive response is received for the `check_payed_method` authentication request.

After getting the added_init_method call, the registrar must confirm ownership of the user's domain using either of the following methods:

- Create the CNAME yamail-secret_name in the domain zone with the value mail.yandex.ru.
- Place a file called secret_name.html in the domain root (it must be accessible via HTTP); the file must contain the secret_value string.

This procedure is performed in effectively the same way as manually verifying a domain, as described here. If the registrar did not generate the MX record when processing payed_url, it can be created, for example, when processing added_init, as specified here.

To call this method, the URL http(s)://registrar.domain.name/added_init_method must be set in the added_init parameter of the save_registrar method.

Just as there is a timeout for check_payed_method, there is also a timeout for added_init. This is why we recommend moving all lengthy operations away from these methods, so you do not block the user web interface.

If the added_init_method request returned an error response, returned nothing, or timed out, the domain will not be connected.

The added_init_method method can be called over HTTP or HTTPS.

```
http(s)://registrar.domain.name/added_init_method ?
domain=<domain name>
 & secret_name=<secret string>
 & secret_value=<secret string>
 & (password=<password> | sign=<signature>)
http(s)://registrar.domain.name/added_init_method ?
domain=<domain name>
 & secret_name=<secret string>
 & secret_value=<secret string>
 & (password=<password> | sign=<signature>)
```

domain	Name of the domain.
secret_name	A secret string that is required for confirming domain ownership.
secret_value	A secret string that is required for confirming domain ownership.
password	The password for accessing the API.
sign	sign = sha1 hash from (domain + secret_name + secret_value + password). Only passed in the HTTP version of the method.

If the request returned an error response, returned nothing, or timed out, the domain will not be connected.

Response format

In response to the method call, the following type of XML structure is returned:

```
<registrar>
  <error>ok|...</error>
</registrar>
```

error*	Method call result. If executed successfully, contains “ok”; otherwise, contains an error message.

Confirmation of domain sign-up

The `added_method` method is used for notifying the registrar of domain connection.

If the domain is connected successfully, the corresponding CNAME and MX records will appear on its NS servers. Yandex notifies the registrar of this by calling the `added_method` method. After the `added_method` call, the registrar gains access to mail import from the domain owner's mailboxes.

Results of `added_method` are ignored; in other words, this method functions as a notification.

The call can be made either over HTTP or HTTPS.

```
http(s)://registrar.domain.name/added_method ?
domain=<domain name>
 & (password=<password> | sign=<signature>)
http(s)://registrar.domain.name/added_method ?
domain=<domain name>
 & (password=<password> | sign=<signature>)
```

`domain`	Name of the domain.
`password`	The password for accessing the API.
`sign`	`sign = sha1(domain + password)`. Only passed in the HTTP version of the method.

Response format

In response to the method call, the following type of XML structure is returned:

```
<registrar>
  <error>ok|...</error>
</registrar>
```

`error`*	Method call result. If executed successfully, contains “ok”; otherwise, contains an error message.

Importing email

The `import_reg_domain` method is used for importing email to Yandex.

Email is automatically imported from its previous storage to Yandex using a POST request.

The request is executed on the registrar's side.

Request parameters:

- `domain` – Domain name.
- `password` – Registrar password that was obtained from the results of the `create_registrar` method.
- `registrar_id` – Registrar ID that was obtained using the `create_registrar` method.
- `mail_proto` – Type of email (IMAP or POP3).
- `use_ssl` – Whether SSL is used on the IMAP/POP3 server:
  0 – Not used.

1 – Used.

- `mail_server` – Domain name of the IMAP/POP3 server where the mail for this domain is hosted.
- `mail_port` – Port. Optional parameter. If omitted, the default port for the protocol is used.
- `e-mails` – XML structure with the names of mailboxes being imported (without the @ symbol and the domain) and the passwords for these mailboxes. The XML looks like this:

```
<emails>
<email><name>name</name><password>password</password>
...
</emails>
```

Response format

In response to the method call, the following type of XML structure is returned:

```
<domains>
  <domain>
    <name>...</name> – domain name
    <emails>
      <email>
        <login>...</login>
        <error>ok|...</error>
      </email>
      ...
    </emails>
  </domain>
  <error_all>ok|...</error_all>
</domains>
```

`name*`	Name of the domain.
`login*`	Name of the mailbox, without the domain part.
`error*`	If the mailbox import task was created successfully, contains “ok”; otherwise, it contains an error message.
`error_all*`	Request results. If the tasks for importing all the mailboxes were created successfully, contains “ok”; otherwise, it contains an error message.

Confirmation of email import

The `transfer_succeed` method is used for notifying the registrar of mail import completion.

The `transfer_succeed` method is called on the Mail for Domains side when the mailbox import is complete.

Results of `transfer_succeed` are ignored; in other words, this method functions as a notification.

This method can be called over HTTP or HTTPS.

```
http(s)://registrar.domain.name/transfer_succeed ?
domain=<domain name>
 & (password=<password> | sign=<signature>)
http(s)://registrar.domain.name/transfer_succeed ?
domain=<domain name>
 & (password=<password> | sign=<signature>)
```

`domain`	Name of the domain.
`password`	The password for accessing the API.
`sign`	`sign = sha1(domain + password)`. Only passed in the HTTP version of the method.

Response format

In response to the method call, the following type of XML structure is returned:

```
<registrar>
  <error>ok|...</error>
</registrar>
```

`error*`	Method call result. If executed successfully, contains “ok”; otherwise, contains an error message.

Notification of domain disconnection

The `delete_method` and `something_delete` methods are used for getting notification when a domain is disconnected.

When a user disconnects a domain from Yandex.Mail, either `delete_method` or `something_delete` is called on the Mail for Domains side. In order to call these methods, the `delete_url` URL must be passed in the `save_registrar` method.

Results of `delete_method` and `something_delete` are ignored; in other words, these methods function as notifications.

The `delete_method` method is called over HTTPS:

```
https://registrar.domain.name/delete_method ?
domain=<domain name>
 & password=<password>
https://registrar.domain.name/delete_method ?
domain=<domain name>
 & password=<password>
```

`domain`	Name of the domain.
`password`	The password for accessing the API.

For making the call over HTTP, the `something_delete` method is used:

```
http://registrar.domain.name/something_delete ?
domain=<domain name>
 & sign=<signature>
http://registrar.domain.name/something_delete ?
domain=<domain name>
 & sign=<signature>
```

`domain`	Name of the domain.
`sign`	`sign = sha1(domain + password).`

Response format

In response to the method call, the following type of XML structure is returned:

```
<registrar>
  <error>ok|...</error>
</registrar>
```

`error*`	Method call result. If executed successfully, contains “ok”; otherwise, contains an error message.

Getting information about the current domain state

The `registrar_check_domain` method is used for getting information about the current state of the domain.

`domain`	Name of the domain.
`registrar_id`	The registrar ID that was obtained using the `create_registrar` method.
`password`	The password for accessing the API.

Response format

In response to the method call, the following type of XML structure is returned:

```
<domains>
  <registrar>
    <name>...</name>
  </registrar>
  <domain>
    <domain_name>...</domain_name>
  </domain>
  <stage>
    added|mx-check|owner-check
  </stage>
  <mx>
    <last-date></last-date>
    <next-date></next-date>
    <mx-found>no|...</mx-found>
  </mx>
  <owner>
    <last-date></last-date>
    <next-date></next-date>
    <sub-domain-name>...</sub-domain-name>
    <sub-domain-exists>ok|no</sub-domain-exists>
    <sub-domain-value>no|...</sub-domain-value>
    <file-url>...</file-url>
    <file-exists>ok|no</file-exists>
    <file-content>...</file-content>
    <file-content-match>ok|no</file-content-match>
  </owner>
  <error>ok|...</error>
</domains>
```

`domain_name*`	Name of the domain.
`stage*`	Domain status: “added” – Activated successfully. “mx-check” – Waiting for verification of MX records. “owner-check” – Waiting for verification of domain ownership.
`mx`	Information about verification of the MX record. This element is present if `stage` shows the domain status “mx-check”.
`last-date*`	Date and time of the previous check.
`next-date*`	Date and time of the next check.
`mx-found*`	Contents of the MX record for the domain being checked. If the record is not found, contains “no”.
`owner`	This element is present if `stage` shows the domain status “owner-check”.
`sub-domain-name*`	

Name of the subdomain that the CNAME record is being checked for.

`sub-domain-value*`

If the subdomain exists, the contents of its CNAME record are shown. If the CNAME record is missing, contains “no”.

`file-url*` Name of the file being checked.

`file-exists*` Results of checking whether the file exists:

“ok” – Exists.

“no” – File not found.

`file-content*` Search string in the file being checked.

`file-content-match*`

Whether the search string is present in the file being checked. Acceptable values:

- "ok" — Search string found.
- "no" — String not found.

`error*` Request results. If the method was called successfully, the element contains “ok”; otherwise, it contains an error message.

Checking the state of mail import

The `registrar_check_import` method is used for getting information about the current state of mail import.

`domain` Name of the domain.

`registrar_id` The registrar ID that was obtained using the `create_registrar` method.

`password` The password for accessing the API.

Response format

In response to the method call, the following type of XML structure is returned:

```
<domains>
  <registrar>
    <name>...</name>
  </registrar>
  <domain>
    <domain_name>...</domain_name>
    <emails>
      <email>
        <login>...</login>
        <imported>...</imported>
        <to-import>...</to-import>
        <error_import>no|ok|...</error_import>
      </email>
      ...
    </emails>
  </domain>
  <error>ok|...</error>
</domains>
```

`domain_name*` Name of the domain.

`login`*	Name of the mailbox, without the domain part.
`imported`	Number of imported messages. This element is present if `error` does not contain an error message.
`to-import`	The number of messages left to import. This element is present if `error` does not contain an error message.
`error_import`*	State of the mail import process: "ok" – Mailbox import completed successfully. "no" – Mailbox import has not started. If the import was performed incorrectly, the element contains an error message.
`error`*	Request results. If the method was called successfully, the element contains "ok"; otherwise, it contains an error message.

Yandex

Yandex.Mail for Domains API

Developer's reference

15.01.2016